Operational Planning for Multiple Heterogeneous Unmanned Aerial Vehicles in Three Dimensions

by

Blair Ellen Leake Negron

B.S. Operations Research
United States Air Force Academy, 2007

Submitted to the Sloan School of Management in
Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN OPERATIONS RESEARCH
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Signature of Author: ______________________
Sloan School of Management
Interdepartmental Program in Operations Research
May 14, 2009

Approved by: ______________________
Stephan E. Kolitz
The Charles Stark Draper Laboratory, Inc.
Technical Supervisor

Certified by: ______________________
Hamsa Balakrishnan
Assistant Professor, Aeronautics and Astronautics and Engineering Systems
Thesis Advisor

Accepted by: ______________________
Cynthia Barnhart
Professor, Civil and Environmental Engineering
Co-Director, Operations Research Center

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **JUN 2009**	2. REPORT TYPE	3. DATES COVERED **00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE **Operational Planning for Multiple Heterogeneous Unmanned Aerial Vehicles in Three Dimensions**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,77 Massachusetts Avenue,Cambridge,MA,02139**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Unmanned aerial vehicles are being incorporated in an increasing variety of operations. To take full advantage of the vehicles, the plans for the operations should integrate each vehicle’s capabilities when planning the operations. This thesis focuses on planning operations for multiple, heterogeneous UAVs for the purpose of monitoring Earth’s phenomena through data collection. The planning is done for flight in three dimensions. The problem also includes time window constraints for data collection and incorporates human input in the planning process. Two solution methods are presented: (1) a mixed-integer program, and (2) an algorithm that utilizes a metaheuristic to generate composite variables for a linear program, called the Composite Operations Planning Algorithm. The suitability of the two methods to solve the operations planning problem is compared based on the ability of each of the methods to find high-value, feasible solutions for large-scale, operationally sized problems in a reasonable amount of time. The analysis shows that the Composite Operations Planning Algorithm can develop operations plans for problems including 15 UAVs and 5000 nodes in less than 25 minutes using a desktop computer.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**135**	

THIS PAGE INTENTIONALLY LEFT BLANK

Operational Planning for Multiple Heterogeneous Unmanned Aerial Vehicles in Three Dimensions

By

Blair Ellen Leake Negron

Submitted to the Sloan School of Management
on May 14, 2009 in partial fulfillment of the requirements for the
Degree Master of Science in Operations Research

ABSTRACT

Unmanned aerial vehicles are being incorporated in an increasing variety of operations. To take full advantage of the vehicles, the plans for the operations should integrate each vehicle's capabilities when planning the operations. This thesis focuses on planning operations for multiple, heterogeneous UAVs for the purpose of monitoring Earth's phenomena through data collection. The planning is done for flight in three dimensions. The problem also includes time window constraints for data collection and incorporates human input in the planning process.

Two solution methods are presented: (1) a mixed-integer program, and (2) an algorithm that utilizes a metaheuristic to generate composite variables for a linear program, called the Composite Operations Planning Algorithm. The suitability of the two methods to solve the operations planning problem is compared based on the ability of each of the methods to find high-value, feasible solutions for large-scale, operationally sized problems in a reasonable amount of time. The analysis shows that the Composite Operations Planning Algorithm can develop operations plans for problems including 15 UAVs and 5000 nodes in less than 25 minutes using a desktop computer.

Technical Supervisor: Stephan Kolitz
Title: Member of the Technical Staff
The Charles Stark Draper Laboratory, Inc.

Thesis Advisor: Hamsa Balakrishnan
Title: Assistant Professor
Aeronautics and Astronautics and Engineering Systems

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

I am very thankful for all of the support that I received while working on this thesis. So many people influenced and inspired me while I completed the work. Thank you!

First and foremost, I want to thank God for providing me with the opportunity to glorify His name through my work here at MIT. He has blessed me beyond measure, Col 3:23.

I would like to thank Dr. Steve Kolitz and Professor Hamsa Balakrishnan for guiding me through the thesis writing process. They provided me with support and advice throughout my two years working on this thesis. I especially want to thank them for letting me work from Texas, even though it was more difficult!

I also want to thank Col. Andrew Armacost for his invaluable support. I wouldn't be here graduating from MIT if it hadn't been for his guidance through my four years at the Academy and his help in getting me accepted to this program. I have yet to find another teacher who is so willing to go out of his way to help students. He is a truly remarkable teacher and mentor.

I would like to thank my friends who helped me get through MIT and made it fun. Mallory and Lisa—thank you for being there; I am so glad that I met both of you! Chris, thanks for explaining everything to me!

Next, I want to thank my family. My mother and father, who always supported me in everything that I do. Thank you for always pushing me to do my best and supporting me no matter what the cost! Victoria, Adam, John, Meredith, Aaron, Perri , and Evan—you all are awesome!

Last, but not least, I want to thank my wonderful, amazing husband, Evin, for his support throughout this process. I know you didn't like living away from me, honey, but I am moving to Texas soon! You are amazing and I am truly blessed to have you in my life. I can't wait to start our next chapter together.

This thesis was prepared at the Charles Stark Draper Laboratory, Inc. under NASA ESTO's AIST-05 program, Grant Number NNX06AG17G.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

Blair L. Negron, 2Lt, USAF May 14, 2009

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

The advancement of technologies for unmanned aerial vehicles (UAVs) has allowed the vehicles to be used in an increasing variety of operations. Current planning for many of the operations do not take full advantage of the capabilities of the vehicles, because of the complexity inherent in the planning of these operations.

The purpose of this thesis is to develop an algorithm to plan UAV operations. The development of the algorithm addresses two main challenges to the UAV operations planning process. The first challenge is generating plans that include multiple, heterogeneous UAVs, and the second is planning the operations for flight in three dimensions.

This chapter will introduce the topics discussed throughout this work. The first section provides an overview of the chapters included in the thesis. The second section summarizes the contributions of this thesis. The third section states the motivation for this work.

1.1 Thesis Overview

This thesis presents and analyzes two methods that can be used to plan UAV operations. The methods are developed through six chapters. An overview of the chapters follows.

Chapter 2 presents the operational concept for the planning of UAV operations. This chapter introduces the Earth Phenomena Observing System (EPOS), a planning and control testbed for the coordination of a system of satellites, UAVs, and unmanned surface vehicles

(USVs). It is shown that the algorithm developed in this thesis can be incorporated into EPOS as the *UAV Planner*, which develops the plans for the UAV portion of the system. The chapter then presents operational uses for the UAV Planner, including the use of the planner to develop operations for UAVs to assist firefighting missions. It is shown that the UAV Planner can be used in either a *centralized system*, in which there is a single command station to monitor the operations of every UAV, or a *decentralized system*, in which multiple command stations control subsets of UAVs. At the end of the chapter, the *UAV Planner Problem* statement describes the constraints for the development of operations plans. A solution to the UAV Planner Problem provides a three dimensional plan of operations for a system of multiple, heterogeneous UAVs.

In **Chapter 3**, the UAV Planner Problem is mathematically formulated. The chapter discusses how to model the constraints presented at the end of Chapter 2. It is shown that the UAV Planner Problem can be formulated as a network problem. Previous literature describing similar problems is reviewed to assist in the development of an algorithm for the UAV Planner Problem. In particular, the *Orienteering Problem* and the *USV Observation-Planning Problem* are presented as problems that closely resemble the UAV Planner Problem. The end of the chapter introduces a mixed integer programming (MIP) model for the UAV Planner Problem. The model is implemented using optimization software. An analysis of this solution method is provided in Chapter 5.

Chapter 4 formulates the *Composite Operations Planning Algorithm* (COPA) to solve the UAV Planner Problem. The chapter begins by introducing full path composites, which link a UAV type with a full path. A reformulation of the UAV Planner Problem using composite variables is given. COPA, which utilizes this reformulation, is then presented. The three steps of the algorithm are introduced: Subset Allocation, Composite Generation, and Full Path Composite Variable Linear Program. The last portion of the chapter focuses on the implementation of COPA, called the COPA software. The implementation is done in Java and is used in the testing and analysis of the program.

The testing and analysis of the MIP and COPA are presented in **Chapter 5**. The chapter begins with a comparison between the implementation of the MIP and the COPA software. It is shown that, although the MIP provides an exact solution, COPA is able to provide a plan of operations quickly. In addition, it is shown that the MIP is unable to produce an operations plan for cases including more than 15 tasks. COPA is able to provide plans of operations for large cases including 500 tasks. The next portion of the analysis presents scenarios to demonstrate COPA's planning capabilities. First, a scenario is used to illustrate how COPA can incorporate a human operator into the planning process. Then, two firefighting scenarios are presented. The scenarios are treated as both centralized and decentralized systems. They demonstrate how COPA generates a plan of operations for the UAVs for both types of systems.

The last chapter, **Chapter 6**, provides a summary of the work presented in this thesis. The chapter summarizes the contributions of this thesis. Proposals for modifications to COPA are presented, including the incorporation of weather data into the planning process. In addition, future work concerning the UAV Planner Problem is suggested. The chapter ends by summarizing the conclusions made in this paper.

1.2 Contributions

In the development of an algorithm to plan UAV operations, the following significant contributions are made through this work.

- **This thesis develops and implements a Mixed-Integer Programming model to solve the UAV Planner Problem.** The solution to the MIP develops plans for operating multiple, heterogeneous UAVs in three dimensions. The MIP is implemented using ILOG's OPL Studio 5.5 and solved using the CPLEX 11.0 solver.

- **The development and implementation of COPA, an algorithm to plan operations for multiple, heterogeneous UAVs for flight in three dimensions.** This algorithm uses a local search heuristic to generate composites which link a full path and UAV type. The composites are then used in a composite variable linear program. The result of the algorithm is used to develop a plan of operations for the system of UAVs. The algorithm is coded in Java, which calls CPLEX 11.0 to solve the linear program.

- **Testing and comparison of the MIP and COPA.** The MIP and COPA are tested and compared to determine which method is better suited to be used to plan UAV operations.

- **Development of operational scenarios to depict the use of COPA.** Two scenarios are developed to illustrate the use of COPA to fight wildfires.

- **Recommendations for modifications to COPA.** Possible modifications within the current framework of COPA could provide increased capabilities in planning UAV operations. For example, including weather data in the planning process.

1.3 Motivation for Thesis

The purpose of this thesis is to develop an algorithm that plans the operations for a set of multiple, heterogeneous UAVs. In addition, the plans should be three-dimensional and satisfy the constraints of the UAV Planner Problem. The goal of this work is to develop an algorithm that can generate the plans quickly, while ensuring that the plans include the collection of valuable data for those deploying the UAVs.

The plans generated by the algorithm will be useful in the development of the planning and control testbed called EPOS. The algorithm can be implemented for use in firefighting missions. In addition, the planning algorithm is developed for used in any operations in which UAVs can make a positive impact.

Chapter 2

Unmanned Aerial Vehicle Earth-Observing System Operational Concept

This chapter introduces the *Earth Phenomena Observing System* (EPOS), with a focus on the UAV component of the system. Draper Laboratory has been developing EPOS under NASA ESTO (Earth Science Technology Office) funding since 2000. EPOS is a closed-loop planning and control testbed for the coordination of a system of satellites, unmanned aerial vehicles (UAVs), and unmanned surface vehicles (USVs) to collect data for the purpose of monitoring Earth phenomena. This chapter presents the purpose and functional structure of EPOS and the potential uses for EPOS, specifically focusing on its UAV capabilities.

The chapter then concentrates on the component of EPOS that plans the operations for multiple heterogeneous UAVs; this component is known as the *UAV Planner*. The role and function of the UAV Planner within EPOS is described.

The next part of the chapter presents previous uses of UAVs to monitor Earth's phenomena, focusing on the use of UAVs in firefighting missions. The section describes two operational concepts for the UAV Planner: a *centralized system*, in which a central command station controls the entire set of UAVs, and a *decentralized system*, in which there are multiple command stations control subsets of the UAVs.

The end of the chapter provides a statement of the operations planning problem. The statement describes aspects of the operational problem that should be considered when creating the operations plan.

2.1 Earth Phenomena Observing System (EPOS)

The Earth Phenomena Observing System (EPOS) is a planning and control test bed for the coordination of a system of satellites, UAVs, and unmanned surface vehicles (USVs). Each component, called a sensor platform, carries data-collecting sensors. The sensor platforms work together to provide scientific data to monitor and study Earth's natural phenomena. This data include aerial images, wind samples, water temperature samples and other metrics that the sensors have the ability to measure. The system can be used to plan the monitoring of multiple phenomena, including hurricanes, flash floods, forest fires, and other natural events. Example 2-1 provides an illustration of how EPOS utilizes its capabilities to track a hurricane.

Example 2-1. *An EPOS concept of operations can be illustrated through a hurricane scenario. In order to monitor a hurricane, EPOS can utilize all sensor platforms. The satellites can provide aerial images of the hurricane to track the movement and location of the storm. USVs can be deployed to collect water temperatures, while UAVs can collect wind speeds at various altitudes, both assisting in the forecast of the storm's trajectory. By enabling these observations, EPOS enhances our ability to understand the hurricane and predict its movements.*

EPOS develops a coordinated operations plan to collect the observations. An *operations plan* specifies a location for each sensor platform at specific times throughout the planning horizon. The first aspect of the operations plan is the location; for UAVs, the location must be defined in three dimensions: latitude, longitude, and altitude. The second aspect of the operations plan is time. The sensor platforms should have sufficient time to travel between locations where they can collect data and time to collect the data once the location has been reached.

One goal of EPOS is to enable the use of sensor platforms as a *sensor web*. A sensor web is a system of spatially distributed sensors that can dynamically respond to its observations. In order to achieve the mission of a sensor web, EPOS must engage in re-planning as a response to the data is gathered by the sensor platforms. This re-planning is known as *dynamic re-planning*, and will be discussed further in the next section.

People are an integral part of the EPOS system. People drive the system's planning by requesting the collection of data and monitoring the plans produced by EPOS. Requests for data can come from multiple interested parties: scientists studying the phenomena, alert and warning centers, or those with a practical interest. For example, firefighters might want to observe the location of a fire to determine how they can stop the spread of the fire. These people, defined as *users* of the EPOS system, ask the system to collect specific pieces of data. Their requests define the observations that the sensor platforms will collect; plans are developed to satisfy these requests. More details on the role of people in the system will be discussed later in this chapter.

2.1.1 Earth Phenomena Observing System Functional Overview

The primary objective of the EPOS functional structure is to create an operations plan for the sensor platforms to collect data. The operations plan introduced in the previous section specifies both movement and data collection for the sensor platforms. This section will describe how EPOS develops an operations plan for each group of sensor platforms. The process has three steps that correspond with the three levels of the diagram in Figure 2.1.

In the first step, the *Observation Coordination Planner* inputs requests for data that originate from the system users. These requests define *tasks* (or *targets*) where sensor platforms are to collect data. The Observation Coordination Planner completes a high level division of the tasks; allocating each task to be completed by a specific type of sensor platform.

In the second step, the operations plan is created independently for each group of sensor platforms. Each group has an independent *platform planner,* which receives a list of tasks assigned its respective sensor platform by the Observation Coordination Planner. The platform planner creates an operations plan for the sensor platforms under its control that direct the platforms to perform tasks on the list. The planner attempts to create an operations plan that is valuable to the users, gathering the most beneficial information about the natural phenomena.

In the last step, the operations plan is executed with the assistance and approval of a human operator. The operator directs the movement of the sensor platforms using the operations plan proposed by the platform planner. In this step, the operator is able to make changes to the proposed plans for the purpose of creating a more valuable plan.

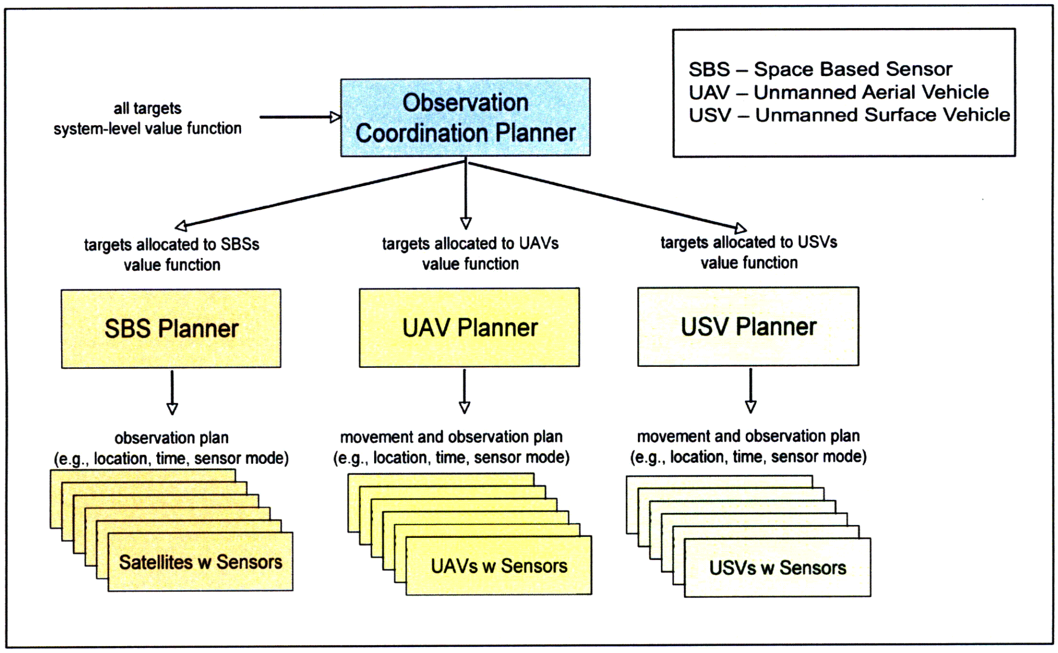

SBS – Space Based Sensor
UAV – Unmanned Aerial Vehicle
USV – Unmanned Surface Vehicle
all targets
system-level value function
Observation Coordination Planner
targets allocated to SBSs
value function
targets allocated to UAVs
value function
targets allocated to USVs
value function
SBS Planner
UAV Planner
USV Planner
observation plan
(e.g., location, time, sensor mode)
movement and observation plan
(e.g., location, time, sensor mode)
movement and observation plan
(e.g., location, time, sensor mode)
Satellites w Sensors
UAVs w Sensors
USVs w Sensors

Figure 2.1 EPOS Functional Overview

Dynamic re-planning and execution is achieved by repeating these steps when necessary. Reasons for re-planning include additions or changes to the users requests, responses to previously collected data, or changes in weather that inhibit the collection of data by sensor platforms. After re-planning is completed, the new operations plan is executed directly from the present state of the system; assets are not required to return to a headquarters or home base before executing the new plan.

2.1.2 *Sensor Web*

As discussed, a goal of EPOS is to enable the use of sensor platforms as a sensor web. A sensor web is a collection of coordinated sensors. The concept of a sensor web was developed at the National Aeronautics and Space Administration (NASA):

> "…the Sensor Web consists of a system of wireless, intra-communicating, spatially distributed sensor pods that can be easily deployed to monitor and explore new environments…"[17]

The sensors have a centralized purpose: to provide valuable scientific data, allowing scientists and researchers to monitor the environment in the area that the sensors are deployed. The information can be exploited and used for a variety of purposes. In many cases, as with EPOS, a sensor web is developed to observe Earth's phenomena.

A sensor web contains both *control nodes* and *sensor nodes*. Control nodes collect data gathered by the system and direct the flow of information throughout the sensor web. Sensor nodes are the sensor platforms that gather observations, which are transmitted to the control nodes. In EPOS, sensor nodes are the sensor platforms: satellites, UAVs, and USVs.

While a system of sensors working together to achieve a common goal is not a new idea, the sensor web utilizes the availability of new technology to improve upon previous concepts. Unlike the systems of "distributed sensors" or "sensors networks," the goal of the sensor web is to dynamically respond to data collected by sensors. Distributed sensor systems and sensor networks contain multiple spatially distributed ground, air, and space resources that collect raw data which is given directly to a centralized control center for processing. The data flows directly from the sensors to the control center. Conversely, the goal of a sensor web is to have data flowing in multiple directions; between sensor nodes to control nodes, between two sensor nodes, or between two control nodes. This allows the sensors to dynamically respond to data flowing from each sensor node. A visual of a sensor web is shown in Figure 2.2.

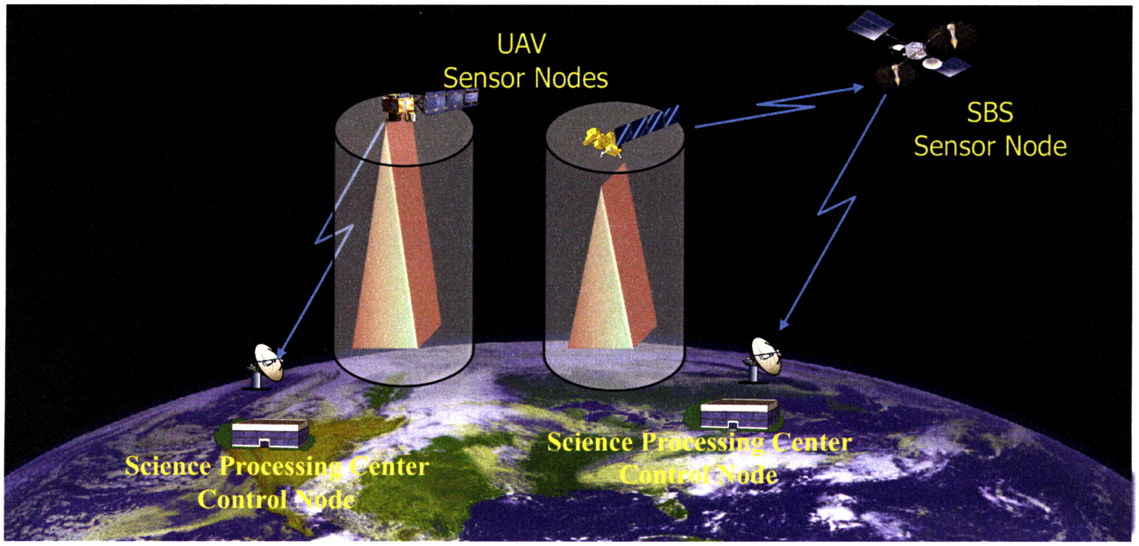


Figure 2.2 Sensor Web Diagram [41]

As discussed above, this response is achieved through dynamic re-planning and execution. The system performs re-planning while continuing current operations. The new plan is developed as an extension of the current plan without creating inefficiency in the operations. When re-planning is completed, the new plan is spliced in the currently executing plan. To achieve this, the new plan must incorporate the placement and activities of the sensor nodes in the current plan when planning future operations.

2.1.3 The Role of the UAV Planner

The primary role of the UAV planner is to plan the operations for the UAV sensor platforms in EPOS. It will be incorporated into the functional structure of EPOS and will serves as part of a control node in the EPOS sensor web.

The UAV Planner is a platform planner, as defined in Section 2.1.1. As a platform planner, the UAV Planner is part of the second stage of planning in EPOS and follows the planning method described for platform planners. The UAV Planner receives a list of tasks from the Coordination Planner. The UAV Planner then creates an operations plan for the system the UAVs. This plan attempts to obtain the most valuable information for those monitoring the phenomena.

In addition, the UAV Planner will be continually receiving data from nodes in the EPOS sensor web. The planner will respond to these inputs by incorporating new information into the UAV operations plan. For example, in the case of using the system to monitor a forest fire, if the control node receives information that the temperature in a particular location has significantly increased, it might plan to obtain an aerial image of the location to determine if the forest fire has spread to the area.

2.1.4 Human Collaboration

As noted above, EPOS includes people in its planning process. The system includes both human *operators* and *users*. It is important to make a distinction between the roles of the human involved with the system. The *operator* monitors the operations planning and ultimately determines the plan that is executed. The *users* submit requests for the system to collect data samples that will be valuable to him or her.

A human operator will direct re-planning and oversee the processes of each of EPOS platform planners, including the UAV Planner. Throughout the operations, the operator of the UAV Planner can see the list of tasks assigned to the UAV system. The operator has the freedom to direct all aspects UAV operations; the individual operators determine the extent of involvement. He can construct the complete operations plan for UAVs, if he desires, or he can execute the plan exactly as prescribed by the UAV Planner. After the planning stage, the human operator executes the selected plan and monitors the UAVs as the plan is executed.

The users of the EPOS system are the scientists, researchers, firefighters, weather analysts, and other users who benefit from the data collected by the system and use the data to monitor Earth's phenomena. They are the driving force behind the operations of EPOS, because they generate the requests for data from which operation plan is developed. They also perform the crucial job of quantifying how "valuable" an observation is to their research or operations. This assigned value is used in determining which tasks are the most critical for the system to perform.

2.2 Operational Scenarios

This section discusses practical uses for EPOS, specifically the system's UAV operations. While EPOS could be used to plan the monitoring of multiple phenomena, this section focuses on use of the system for fighting fires. Previous instances of the use of UAVs to supplement firefighting efforts are introduced; as well as current studies in this area. The section then presents the proposed operational concept for the UAV Planner in future firefighting scenarios.

Forest fires are a hazardous natural phenomenon. Each year, wildfires destroy millions of hectares of land spread throughout all parts of the world. In addition to the destruction of land, wildfires impact the surrounding community by endangering lives and creating a burden on the local economy to reconstruct infrastructure and homes destroyed by the fire [14]. UAVs provide firefighters with enhanced capabilities to strategically and intelligently fight fires, greatly reducing the amount of land destroyed and its impact on the surrounding community.

2.2.1 Current Studies and Operations

Research into the use of UAVs to assist fighting wildfires has been quickly increasing as UAVs become more affordable and available. NASA and the United States Forest Service (USFS) have worked together to conduct yearly demonstrations of the use of UAVs to fight wildfires since June 2006.

Demonstrations have been conducted with a single UAV. In 2007, the UAV route was built around a pre-determined backbone. The UAV flew along the backbone, deviating only to collect necessary data. The entire route was prepared before take-off and then followed throughout the flight. A map illustrating a route from 2007 is shown in Figure 2.3.

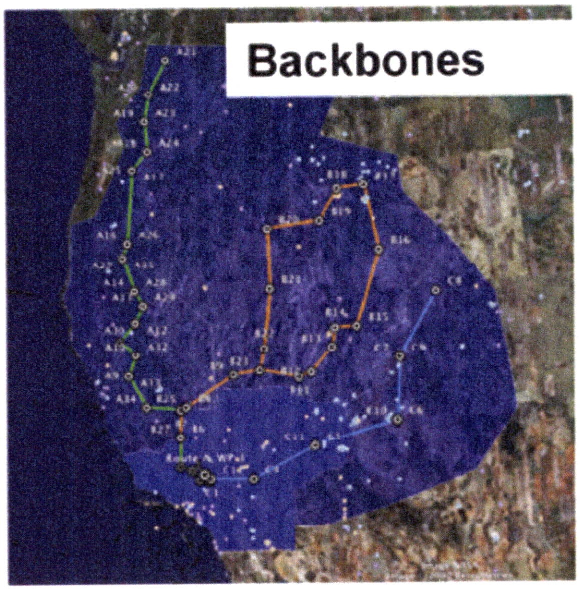


Figure 2.3: UAV Route Backbones for NASA/USFS Fire Demonstrations [40]

In October 2007, NASA deployed the Predator-B Ikhana UAV to obtain images of wildfires in California. It was the first time that the UAV was used in an operational firefighting mission. The firefighters involved in the mission indicated that the information provided by the UAV contributed to their ability to make decisions in planning the mission. After the fires were contained, Ed Hollenshead, the Director of Fire and Aviation for the Forest Service in California, said:

> "This technology is going to be an excellent tool for our Incident Commanders. Its ability to give us real time information and reduce the risk to firefighting crews on the ground is invaluable" [31].

Since this innovative use of UAVs in fighting wildfires, NASA has continued to work to improve this technology in order to further assist firefighters.

Figure 3.4 is an image taken in October 2007 by the Ikhana aircraft over the San Bernardino Mountains of Southern California. The image was processed at NASA Ames Research Center, where it was overlaid on Google Earth maps. The yellow is the active fire. Dark red and purple hues indicate areas where the fire has already burned the vegetation, while the green indicates area that has not been affected by the fire [32].

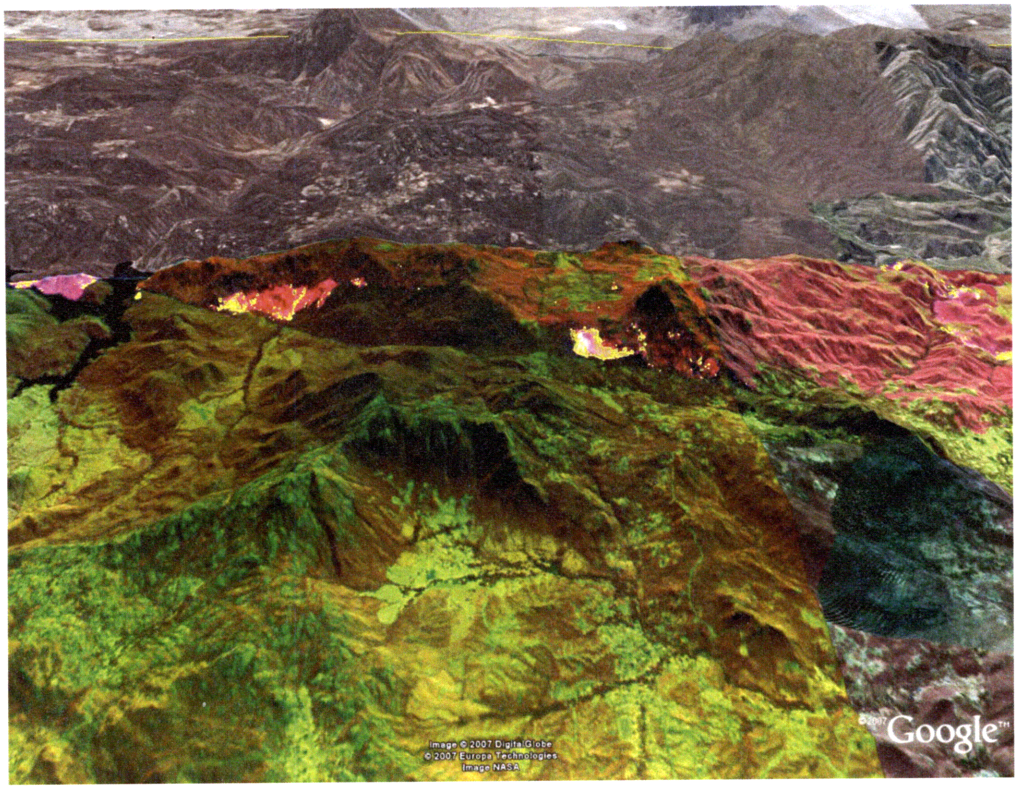


Figure 2.4 Ikhana Image of a California Fire Overlaid on Google Earth Maps [32]

2.2.2 *Projected Issues with Future UAV Use*

The current process for operations planning used by NASA and the Forest Service will need to be altered as new technologies and multiple UAVs are incorporated into firefighting missions. This section will focus on some of the advances that will affect operations planning for UAVs in the future.

As UAVs become less expensive, more than one UAV might be deployed in support of a firefighting mission. To take advantage of the increased capabilities of multiple UAVs, each UAV should have a unique portion of the operations plan. While flying on multiple backbones could provide each UAV with a unique plan, it may be more efficient to fly point-to-point between tasks. An improved UAV planner should have the ability to plan efficient operations for more than one UAV.

In addition, new UAVs might not be identical to the Predator-B that has been used for firefighting operations. This further increases the complexity of the problem, as different UAVs provide different capabilities; they will move at different speeds and have different limitations. These capabilities should be incorporated into the operations planning.

To benefit from these advances in UAV technology, there exists a need for an improved planning capability that takes into account multiple, heterogeneous UAVs operating in a three dimensional environment. The planning capability should fully utilize the resources of the system to provide the maximum amount of information to the users. Ultimately, it will increase the effectiveness of our firefighters, hurricane monitors, and other natural phenomena researchers.

2.2.3 *Concept of Operations using the UAV Planner*

The purpose of this section is to outline the steps to be taken if the UAV Planner is used in a firefighting missions. The first part of the section illustrates how tasks are input by the user and then assigned to the UAV Planner. Then, two concepts of operations are introduced – a *centralized system* and a *decentralized system*. In the centralized system, an operator at a single command center controls all of the UAVs. In the decentralized system, incident commanders maintain control over a subset of UAVs independent from other commanders.

2.2.3.1 Request Inputs

In the first phase of operations for the UAV Planner, users determine the data to be collected by the UAVs. In this example, the users are firefighters. Two examples of tasks are presented: The first is to take an aerial image, and the second is to collect a wind speed sample.

First, the firefighters determine that they would like an aerial image of the north edge of the fire. Using a graphical user interface, the firefighters will indicate the area of which they would like an image. A pair of polar coordinates indicating two corners will demarcate the area of the requested image. The firefighters will also indicate the value of the image to their operations. If the image is critical to the planning of their operations, this task is assigned a very high value. However, if the image is merely 'nice to have' it is indicated by a low value associated with the request.

Second, the firefighters decide that they would like to know the wind speed at a certain altitude over a given area of operations. Similar to the aerial image, the firefighters will determine the area and altitude from which they would like to collect the wind speed. The firefighters will determine the value of the wind speed to their operations. They will input the area of operations, the altitude, and the value of the data into the UAV Planner.

Additional tasks can be requested. After the task requests are collected, the centralized and decentralized systems operate differently.

2.2.3.2 The Centralized System

In a centralized system, an operator at a single command station location utilizes the UAV Planner and dispatches the UAVs according to a selected plan. In this system, the operator has control over the entire system of UAVs. When the UAV Planner is utilized within the EPOS functional structure, it is a centralized system.

The operator takes the following steps to execute the operations plan for the UAVs. First, the operator inputs the task list created by the Coordination Planner into the UAV Planner. Using this task list, the planner creates an operations plan for each UAV in the system. Next, the operator has the opportunity to make changes to the proposed plan. If the operator sees no improvement to the plan, he executes the operations by dispatching the UAVs. Otherwise, he can alter the plan as desired before execution. As the plan executed, the operator monitors the system and can make adjustments to the executing plan as necessary.

2.2.3.3 The Decentralized System

In a decentralized system the UAVs are in separately controlled groups with separate command stations. These command stations are closely linked to incident groups of firefighters on the ground. Note that this system does not take advantage of the EPOS functional structure; in a decentralized system, there is no Coordination Planner to assign tasks to the system of UAVs.

The tasks for the UAVs to accomplish come from the incident group of firefighters. The command station operator inputs the tasks generated by the group. The operator then creates an operations plan for the set of UAVs under his control. The operator approves and modifies the plan as necessary and dispatches the UAVs under his control. He has no information about the operations of the other command stations. Figure 2.5 illustrates a decentralized system.

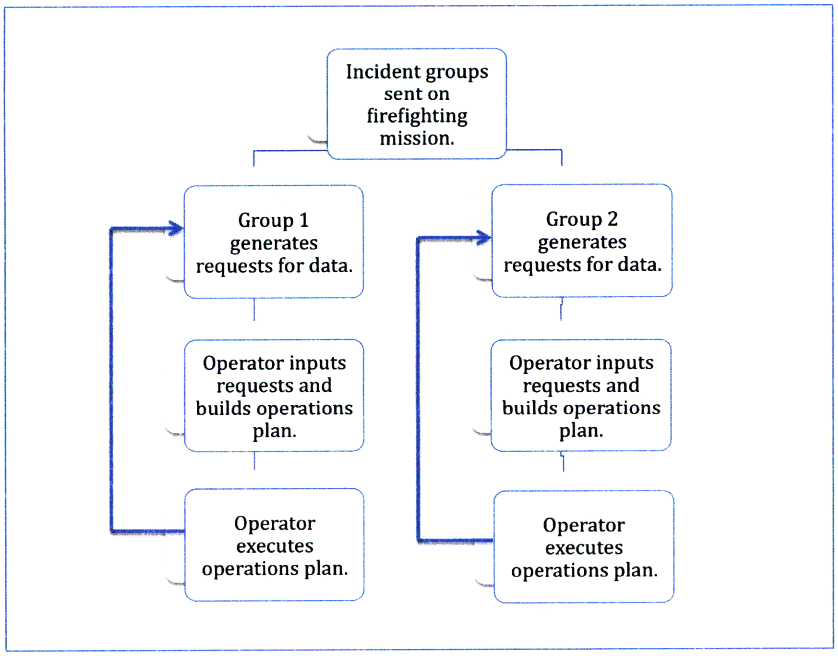

Incident groups sent on firefighting mission.
Group 1 generates requests for data.
Operator inputs requests and builds operations plan.
Operator executes operations plan.
Group 2 generates requests for data.
Operator inputs requests and builds operations plan.
Operator executes operations plan.

Figure 2.5 Functional Structure of a Decentralized System

2.3 UAV Planner Problem Statement

This section discusses the UAV Planner and the operational problem that is addressed; this problem is called the *UAV Planner Problem*. First, the scope and purpose of the problem are presented. Then, the output of the problem—the operations plan—is shown to have two parts: a path plan and an observation plan. The last section discusses characteristics of the problem that should be incorporated into the operations planning.

2.3.1 Scope of Problem

Throughout the operational planning process high-level, mid-level, and low-level decisions are to be made. This section describes each level. The UAV Planner focuses on the mid-level decisions; therefore, these are the decisions addressed in the description of the operational problem of the UAV Planner.

High-level decisions are out of the scope of this research. However, these decisions are critical to the operations of the UAV Planner. High-level decisions focus on decisions that affect the entire EPOS system. They include:

(1) Should UAVs be deployed in support of a mission?
(2) How should the Coordination Planner allocate tasks?
(3) How valuable is a specific observation?

These decisions are made prior to the use of the UAV Planner.

The UAV Planner addresses mid-level decisions, focusing on the construction of an operations plan for the system of UAVs. It assumes that the tasks are known, and that possible locations to complete the tasks and their value to the user are also known. This research focuses on the following questions of different characteristics:

(1) Which tasks should be performed?
(2) In what order should the tasks be performed?
(3) At what time should a UAV perform the tasks assigned to it?
(4) Which UAV should perform which task?

Additional topics and decisions are discussed throughout this paper; however these are the types of questions that will be addressed by this study.

Low-level decisions are also out of the scope of this problem. An example of a low-level decision is to determine the exact trajectory taken by each UAV.

2.3.2 *Purpose of UAV Planner Problem*

The UAV Planner problem has two goals: (1) To develop a process to create a plan of operations for the UAVs which is valuable to the user, and (2) To created the operations plan in a timely manner. These goals must be balanced when developing a method to solve the problem.

The first goal is to create an operations plan for the UAVs that is the most valuable to the user. While the planned movements should be sensible and efficient, the primary objective focuses on the value of data to the user. The operations plan will be presented in more detail in the following section.

The second goal is to be able to construct the routes in a short amount of time. A quick operations planner is essential for the dynamic re-planning and execution needed in a sensor web. First, the operations plan should be determined quickly enough for the plan to be executed; if the operations plan is not completed until after the beginning of the first time period, then the plan is useless and must be reconstructed. In addition, a quick operations planner allows the operator to re-plan if needed; such as in the event that a very valuable task with a short suspense time is requested.

Most likely, there will be a trade-off of these two aspects: a very fast operations planner typically creates less valuable plan; while slower planner may be able find the most valuable operations plan. The emphasis of the UAV Planner Problem is to develop a planner that is practical and can be used in an operational system. For this reason, the planner must be able to work quickly even though it might not produce the most valuable operations plan.

2.3.3 *Operations Plan*

The output of the UAV Planner is an operations plan for the system of UAVs under its control. The plan will have two parts: the first part is in the form of paths that direct each UAV to perform tasks along its respective path; the second part defines the time at which a UAV should be at specific points along the path. The two parts of operations plan are called the path plan and the observation plan.

The *path plan* is an ordered sequence of tasks that create a path. Because each task has an associated location (latitude, longitude, and altitude), the sequence of tasks is fundamentally a sequence of locations that create a route for the UAV to travel along. The path plan defines which UAV will perform which route. The UAV Planner does not determine the exact trajectory between the tasks.

The second part of the operations plan is the *observation plan*. The observation plan defines the times that each UAV arrives and departs each task location along its route. This part is important, as many of the tasks are time sensitive and should be completed within a specified range of time. The significance of time is explored further in the section *Route Characteristics*.

2.3.4 *Multiple Heterogeneous UAVs*

As discussed in Section 2.2, the UAV Planner is developed to provide the capability to create operation plans that incorporate multiple UAVs. This capability requires for the planner to coordinate the path and observation plans in such a way that the user is provided with the most valuable data to understand the natural phenomena.

In addition, the UAV Planner is specifically designed to work with UAVs of different types. Therefore, it is developed with the capability to coordinate the operations plans of multiple UAVs of different types and can be called a *multi-heterogeneous* UAV Planner.

Initially, the type of UAV is not assigned to perform a specific task. Therefore, in order to provide this capability, the UAV Planner has to determine which type of UAV should perform which tasks, taking into account the unique abilities of the specific UAV type.

2.3.5 *Operations Plan Characteristics*

The operations plan takes into account the characteristics of the operational problem. This section presents three characteristics of the operational problem to consider when generating operations plans for the UAVs: three dimensionality, time sensitivity, and continuous operations.

2.3.5.1 Three Dimensionality

The operational problem presented to the UAV Planner is three-dimensional. UAVs have the ability to move in three dimensions, changing altitude in addition to latitude and longitude.

Furthermore, data collection can occur from more than one location; in fact, there are an infinite number of locations that a UAV could collect certain pieces of data. For example, taking an aerial picture of a location on the ground can be done from a wide range of altitudes and angles. The operations plan must choose the exact location (latitude, longitude, and altitude) that a UAV should collect data. The UAV Planner will have to determine how to take advantage of this flexibility.

However, to provide useful operations plans, the UAV Planner should be able to determine restrictions on the location at which a picture could be taken. These restrictions can be attributed to a variety of circumstances. For example, there are operational restrictions. If the user necessitates a specific picture definition or field of view then the locations from which the sensor on the UAV can achieve the specifications is limited. In addition, weather aspects may restrict the range of locations available to complete a task. For example, if the sun is casting large shadows across the main object of an image, the image may be rendered useless to the user and a waste of time for the UAV system. Alternatively, cloud coverage may block the view of an object, again creating a useless image.

2.3.5.2 Time Sensitivity

The nature of Earth's phenomena presents the UAV Planner with an operational problem that is time sensitive. Phenomenon can change quickly and timely data is needed to track their progression. If data is not collected within a specific length of time, then it might be useless to the user. For example, in tracking the winds of a hurricane, if the UAV visits a location after the hurricane has already moved out of the area, then the sample is of no use to the scientist. Therefore, it is critical to take into account the time at which the data is collected.

Example 2-2. *An example of time sensitivity can be found in the firefighting scenario. Perhaps the firefighters would like to know the direction and speed of the wind in order to determine the most likely direction in which the fire will spread. This data would need to be collected quickly, because it affects how the firefighters choose to plan their firefighting operations, as they will send a squad to the area in which the fire is the most likely to spread. However, if the data is not collected in a timely manner, the decision* *will still have to be made. Once the information has been collected, it might be too late for the firefighters to incorporate the information into their decision-making process.*

2.3.5.3 Continuous Operations

A challenge of the UAV Planner Problem is to effectively create and execute operation plans. In order to have continues operations, re-planning will be executed during the current execution. In addition, the new operations plan must follow directly from the current plan. Therefore, the new plan must anticipate where the UAV will be at the time that the new plan is executed. Otherwise, there will be discontinuity between the plans as the UAVs are positioned to execute the new plan.

Also, the UAV Planner does not want the UAV returning to the ground at the conclusion of each plan. It is efficient for the UAVs to return to the base only when necessary to refuel or for maintenance issues.

Chapter 3

Model Formulation and Development

The previous chapter presented the operational characteristics of the UAV Planner Problem. The UAV Planner Problem, as defined in the last chapter, is to create an operations plan for multiple, heterogeneous UAVs. This chapter develops a mathematical representation that addresses the UAV Planner Problem. The mathematical representation begins with a description of the mathematical structure of the UAV Planner Problem, including the definition of terms to describe components of the model. A mathematical representation of the inputs and outputs is also presented, followed by a discussion of the role of the human operator in the planning process.

The mathematical formulation allows us to identify similar problems in the literature. In particular, previous studies on the *Traveling Salesman Problem* assist in understanding UAV Planner Problem and provide insight into methods for solving the problem. The *Orienteering Problem* is introduced as a variant of the Traveling Salesman Problem with characteristics similar to the UAV Planner Problem [23]. In addition, similarities between the UAV Planner Problem and the *Unmanned Surface Vehicle Observation-Planning Problem,* studied by Miller, are discussed [30].

At the end of the chapter, a *mixed integer programming (MIP)* model of the UAV Planner Problem is presented. This MIP utilizes binary and continuous decision variables to build an operations plan for a set of UAVs that satisfies the constraints of the UAV Planner Problem.

3.1 Mathematical Structure of the UAV Problem

This section presents the mathematical model of the UAV Planner Problem. The terminology used to describe the components of the UAV Planner Problem is defined. The inputs and outputs of the model are described. Finally, it is shown how the human operator is incorporated into the operations planning process.

3.1.1 Tasks and Locations

The concept of a *task* was presented in the previous chapter. When users request for the collection of data, they are "tasking" the system to collect data. The concept of a task is broad and dependent on the sensors attached to the UAV. A *location* is the latitude, longitude, and altitude coordinates at which the task can be performed. Occasionally, the term *task location* is also used to describe a specific location at which a task can be performed.

Tasks can be of two types: an *area task* or a *point task. Area tasks* define an area in which the UAV should be for a given time period. These tasks include requests for a UAV to loiter in a given area, to monitor a phenomenon or to be ready to take a nearby aerial image when the time arises. *Point tasks* are tasks that can be performed at a single location. These tasks include collecting the wind speed and obtaining an aerial image.

As previously discussed, each task can be performed at one of multiple locations where it is possible to collect the requested data. To be able to create a mathematical model, the number of possible locations per task will be finite. Increasing the number of possible locations for each task greatly increases the size of the problem. Figure 3.1 displays the concept of having multiple locations for each task.

While allowing tasks to be performed at multiple locations increases the complexity of the problem, it allows for flexibility in the path planning and allows the planner to take advantage of the three-dimensional capabilities of the UAVs. For example, a UAV might not be able to reach that location directly above the area for a requested image; however, it might be able to obtain a picture at an angle that is valuable to the user. Therefore, multiple locations increase the flexibility of the path plans and might result in operations that are more valuable to the user.

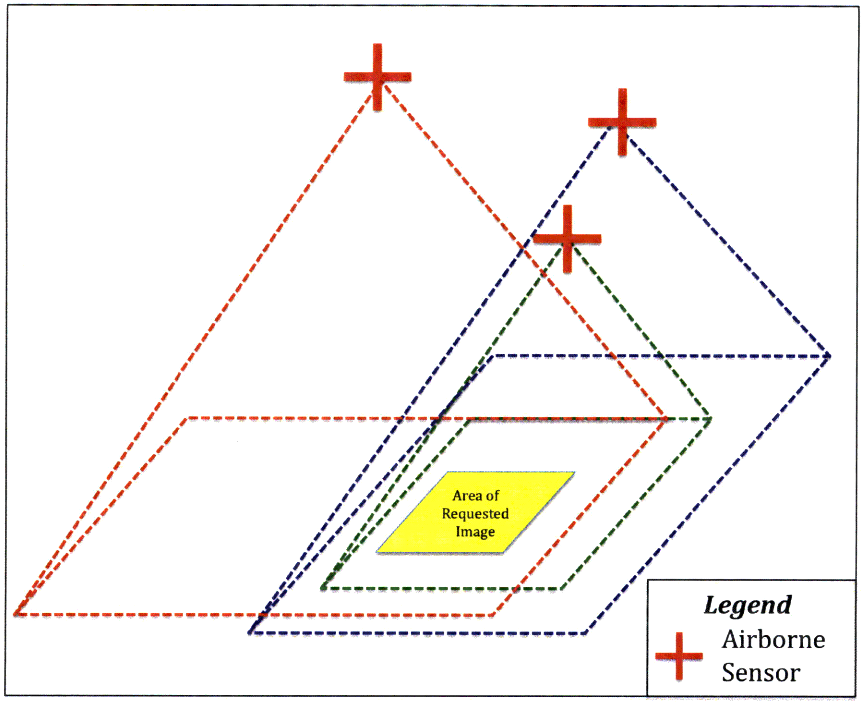


Figure 3.1: Multiple Locations to Image Area on Ground

In addition, for tasks that include aerial imaging, the altitude at which the image is taken determines the resolution of the image. If an aerial imaging task can be taken at either a high or a low altitude, the image taken from the higher altitude will have a lower resolution than the image taken at a lower altitude. In addition, an image taken at a higher altitude will have a larger field of view than an image taken from a lower altitude. By determining which altitude to take the image from, the UAV Planner Problem also determines the resolution and area of the image.

3.1.2 Input

There are two classes of input data for the mathematical model. The first type of input data is data that describes the tasks for the UAVs. The second type is data that describes each UAV's performance specifications.

The time windows for each task are part of the input. To simplify the representation of time, absolute time windows will be converted into relative time with the start time of the operations plan as time 0. For example, consider a task with an absolute time window beginning at 9:30 that will be part of a plan with a start time at 8:00. The early time window will be input as 1.5, because 9:30 is 1.5 hours after 8:00.

The following list describes the input data for each task:

1. *Task Locations*: The set of latitudes, longitudes, and altitudes at which a task could be performed.
2. *Observation Time*: The time it takes for the UAV to complete the task (in hours).
3. *Value*: A measure of how valuable the data collected in the task is to the user. This is used for the objective function, the total value of all completed tasks. The value can be (and usually is) different for different task locations.
4. *Time Windows*: The start and end time during which the data collection task has value. For the model, the absolute time windows will be converted into the number of hours after start time. For example, an early time window of 1.5 indicates that the task must be performed more than an hour and a half from the start time for the plan (time 0).

The following list describes the performance input data for each UAV:

1. *Endurance*: The maximum flight time for each UAV.
2. *Speed*: How fast the UAV moves over the ground. The model assumes speed to be constant when traveling between tasks.
3. *Climb/Sink Rate*: How fast the UAV can increase/decrease its altitude.
4. *Ceiling/Floor*: Operational constraints on the (upper/lower) altitude limits for the UAV.

These inputs are used to calculate other data in the model. For example, the model calculates the time necessary to travel from one location to another for each UAV. Calculation of this value uses both task data and the UAV speed.

3.1.3 Output

As mentioned in the introduction to this chapter, the model outputs an operations plan for each UAV, in the form of a route with times defining when the UAV should be at each location. The path plan is described by the following characteristics:

1. *Task Order:* The order in which the UAV will complete its assigned tasks.
2. *Location Selection:* The path plan will determine the location at which the task will be completed.

The observation plan includes the following:

1. *Start Times:* The time at which the UAV will begin each task assigned to it.
2. *End Times:* The time at which the UAV will complete each task assigned to it.
3. *Departure/Return Times:* The time at which the UAV will depart its command station and the time at which it will return to its command station.

Example 3-1 provides an example of an operations plan containing a path and an observation plan.

Example 3-1. *This example describes an operations plan, including a path plan and an observation plan, given a set of five tasks, $\{t_1, t_2, \ldots, t_5\}$, to be completed by a set of two UAVs, $\{u_1, u_2\}$. The following is a path plan for the example:*

1. *The first UAV, u_1, will complete the ordered set of tasks: $\{t_5, t_1, t_4\}$*
2. *The second UAV, u_2, will complete the ordered set of tasks: $\{t_2, t_3\}$*

The observation plan provides the time that the UAVs will be at each task in their respective assigned tasks. The times given are the number of hours after the execution start time for the plan. The following provides an observation plan for the example:

The first UAV, u_1, departs the command station at time 0, and then perform the following observations. The UAV returns to the command station directly after competing task 4:

Task	***Start Time***	***End Time***
t_5	*0.2*	*0.35*
t_1	*0.45*	*0.6*
t_4	*0.77*	*0.8*

Table 2.1 Example 3-1 Observation Plan for UAV 1

The second UAV, u_2, also departs the command station at time 0, and then perform the following observations. The UAV returns to the command station directly after completing task 3:

Task	***Start Time***	***End Time***
t_2	*0.56*	*0.71*
t_3	*0.89*	*0.92*

Table 2.2 Example 3-1 Observation Plan for UAV 2

This path and observation plans for both UAVs are illustrated below in Figure 3.2. With both the path plan and the observation plan, a complete operations plan is created for each UAV.

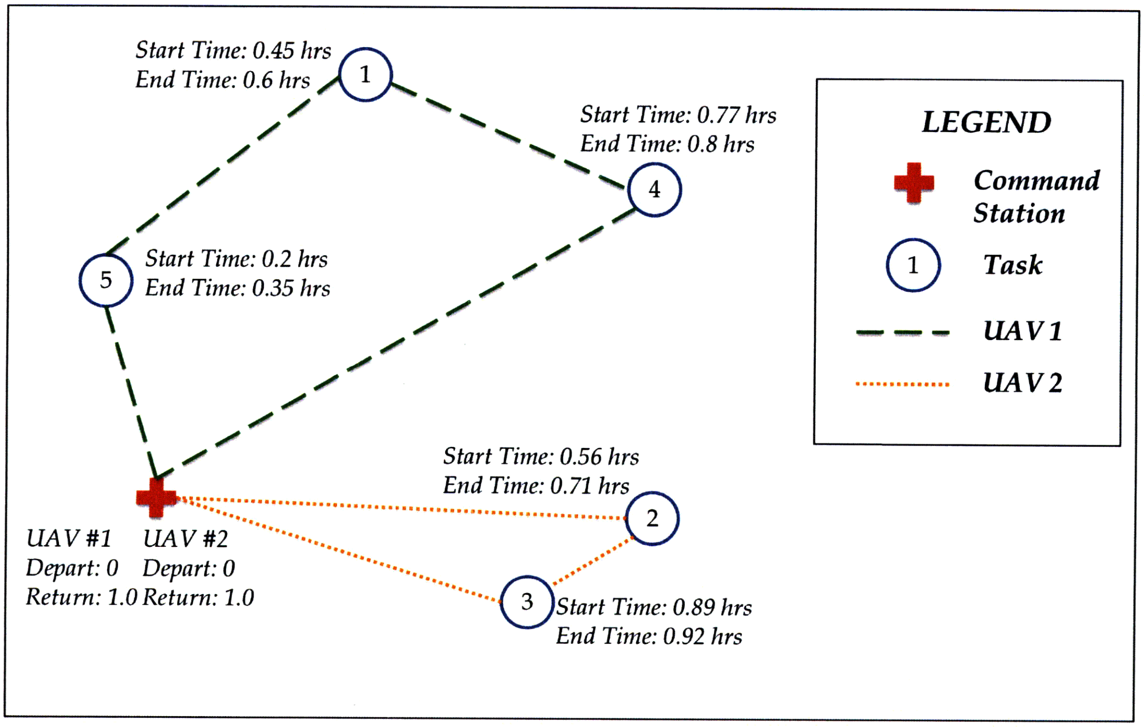


Figure 3.2 Example Operations Plan

3.1.4 Operator Input

As discussed in the previous chapter, the UAV Planner has an *operator* who oversees the operations of the UAVs, and, ultimately directs the UAVs in the operations plan. The operator is also part of the operations planning process for the UAVs. While the incorporation of a human operator into the decision making process might result in the selection of sub-optimal plans, he or she serves as a check on the plans proposed by the UAV Planner for any mistakes, potentially hazardous situations, or possibly to add last minute requests.

The operator can enter the decision making process at two points. First, he or she can assign specific tasks to be completed by a specific type of UAV. Second, the operator can decide which plan will be executed by the UAVs. This is discussed further in the following paragraphs and in Example 3-2.

The human operator can first enter the decision making process before the UAV Planner is executed. In this stage, the human operator communicates to the planner that a specific type of UAV should perform certain tasks. The reason for having human interaction at this point is two-fold. First, the human operator might have a particular bias toward the type of UAV to perform the task. For example, the user who requested the data collected by the task might also request for a specific type of UAV; alternatively, as the operator gains experience he or she might be able to identify the UAV that would be most suited to the task. Second, if the characteristics of a task identify which UAV should perform the task, the operator can assign the UAV to the task. For example, if the sensor necessary to complete the task is only attached to a single type of UAV, then the operator can input for the type of UAV to perform the task. This results in the UAV Planner creating plans more quickly, because it no longer has to consider which UAV type at assign to the task.

At the end of the UAV Planner execution, the human operator selects which operations plan will be executed by the system of UAVs. The operator is presented with multiple sets high value plans; the set of plan with the highest overall value is highlighted. This provides the operator with the ability to select lower value plans for reasons unknown to the UAV Planner, such as the unexpected presence of low-altitude clouds in an area, which would render an aerial image useless.

Example 3-2. *This example illustrate how the operator enters the UAV Planner's operations planning process for a system with six tasks and a two UAV's. The operator enters the process at two points: (1) Before the operations plan is created by the UAV Planner, the operator can assign specific tasks to be performed by specific UAV types; and (2) After the UAV Planner has created operations plans, the operator can choose which plan he would like to execute.*

In this example, before the operations plan is created, the operator indicates that he would like Tasks 1 and 3 to be completed by UAV 1. This is shown in the left side of Figure 3.3. The UAV Planner then creates multiple operations plans that include the operator's input. This is shown on the right side of Figure 3.3. At this point, the operator enters the decision making process for the second time. He can decide to execute either Option 1 or Option 2. Notice that although both options are different, in both operations plans Tasks 1 and 3 are performed by UAV 1.

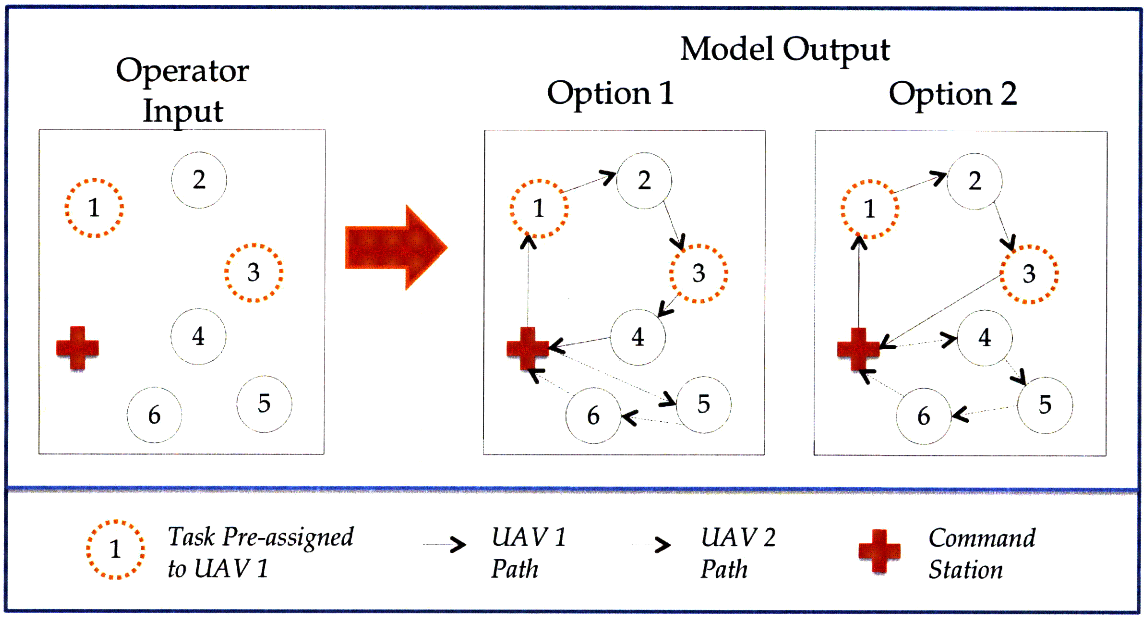


Figure 3.3 Human Collaboration Example

3.2 Network Representation

This section describes the formulation of the UAV Planner Problem as a *network problem*. The class of network problems is defined by the ability to represent the problem on a graph. A graph, *G* = *(N, A)*, is defined as a set of *nodes*, *N*, and *arcs*, *A*. This formulation of the problem will provide insight into methods to solve the problem.

3.2.1 *Network Formulation*

In the UAV Planner Problem, the task locations define the locations of the network nodes; therefore, the set *N* is the set of task locations. The flight paths traveled by the UAV between task locations create the arcs of the network. An example of the UAV Planner Problem as a network is shown in Figure 3.4.

Feasible operations plans follow the arcs and nodes of the network; there are additional guidelines that must be followed in order for the plan to be feasible. A *walk* on a network is defined as an ordered set of nodes, $\{i_1, i_2, \ldots, i_t\}$, such that that the arc $(i_k, i_{k+1}) \in A$ for $k = 1..t$. A *path* is a walk in which no node appears twice in the set. Because we do not want to perform the same task twice, a path is needed. However, if the vehicle returns to its starting location, it then creates a *tour* or a *route*, where $i_1 = i_t$.

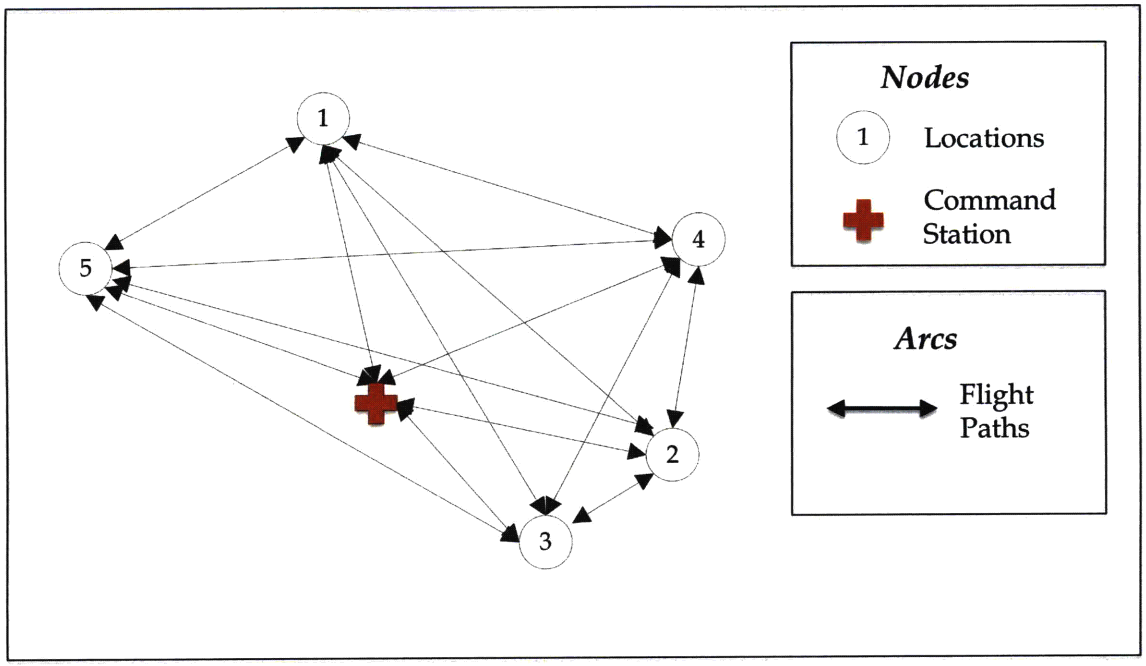


Figure 3.4 A Network Created for an Instance of the UAV Planner Problem

Therefore, the operations plan includes a *path* if the UAV does not return to its command station, or a *route* if it does. Because the graph represents a transportation network and the objective is to find paths or route for vehicles in the network, the UAV Planner Problem can be classified as a *vehicle routing problem*. This will be important in identifying similar problems in the literature.

3.2.2 *Static Graph Representation*

A *static graph representation* of the problem can be shown by a representation of the network formulation of the UAV Planner Problem. Operationally, the operations plan will be performed over a time period; however, the static graph representation captures the plan without a portrayal of time.

Using the same representations from the network formulation of the problem, the nodes represent the locations at which the UAVs perform tasks. The arcs between the nodes illustrate how the UAV will travel between tasks. Figure 3.5 is a static graph representation for the operations of a single UAV.

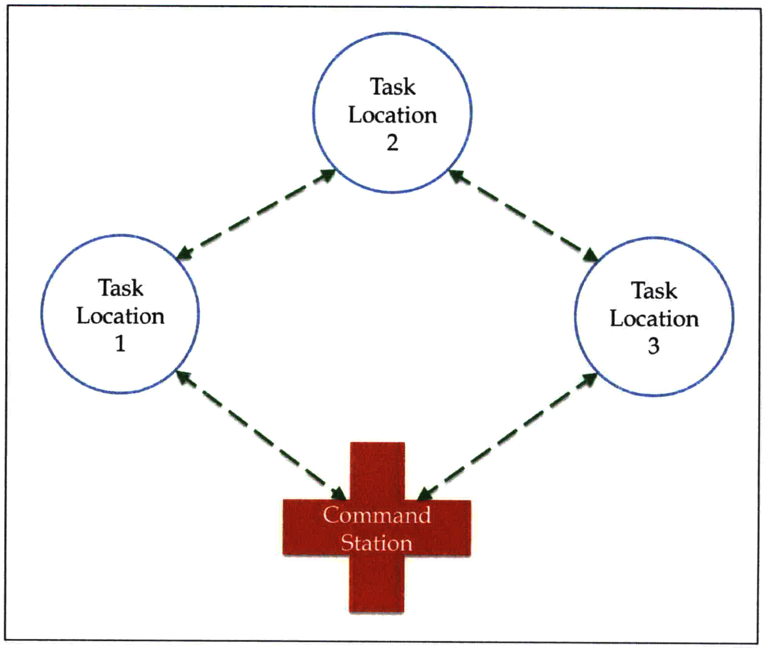


Figure 3.5 Static Graph Representation

3.2.3 *Time-Space Graph Representation*

The static graph representation presented in the previous section is unable to model the time dimension of the UAV's operations. To model position and time, the *time-space representation* can incorporate time by creating nodes that indicate the location of the UAV and the time that the UAV is at the location. Therefore, a node must be created for each task location at every time over the planning horizon. The number of nodes in the representation is very large when illustrating multiple UAVs and tasks. Figure 3.6 provides a time-space representation for the operations of a single UAV.

The time-space representation will assist in scheduling UAV operations, because it records the location of every UAV at each point in time. In addition, the time-space representation assists in dynamic re-planning for the system of UAVs. A new plan will aim to follow directly from the current plan of operations without interruption, such as the need to reposition. The placement of each UAV at the time that the new plan will take over can be found in the time-space representation and used to re-plan operations.

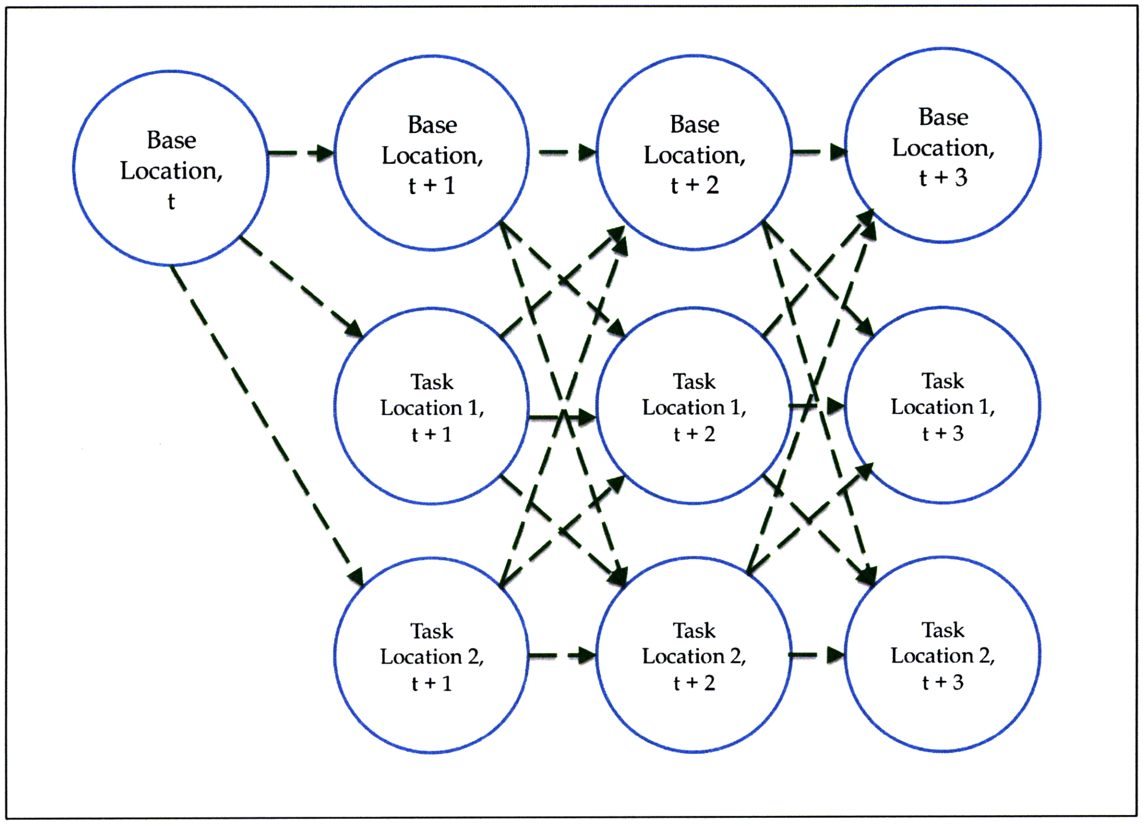


Figure 3.6 Time-Space Representation

3.3 Literature Review

The UAV Planner Problem is fundamentally a Traveling Salesman Problem. This section explores the Traveling Salesman Problem, focusing on available methods to solve the problem. It also introduces three variants of the Traveling Salesman Problem that are similar to the UAV Planner Problem: The *Traveling Salesman Problem with Time Windows*, the *Prize Collecting Traveling Salesman Problem*, and the *Orienteering Problem*.

3.3.1 The Traveling Salesman Problem

A solution to the *Traveling Salesman Problem* is the route with the minimum distance that visits each node in the network. Dantzig describes the problem as following: "Find the shortest route (tour) for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the original point of departure" [16]. This problem is known to be *NP-complete*; therefore, it is classified as a "hard" problem [9]. Because of its difficulty, multiple techniques have been developed in order to solve the problem. Three of the most common methods include integer programming, dynamic programming, and heuristics.

3.3.1.1 Computational Complexity of the Traveling Salesman Problem

An algorithm can solve in *polynomial time* if the function of the runtime to guarantee a solution is polynomial with respect to the size of the problem. More specifically, the runtime of an algorithm can be modeled by a function *f(n)* where *n* denotes the number of nodes in the network. If the function *f(n)* is of the form Kn^c where *K* and *c* are both constants, then the algorithm is of polynomial time [1]. This algorithm would then be classified as "good." However, if the worst case of the algorithm is not of polynomial time, then it is a "bad" algorithm [29].

In literature, problems are classified as either "hard" or "easy" depending on whether or not there exists an algorithm that can find the solution in polynomial time. Currently, no known algorithms for the Traveling Salesman Problem satisfy this criterion. For this reason, the Traveling Salesman Problem is classified as a "hard" problem. In addition, it is an *NP-complete* problem, because the best-known algorithm has a non-polynomial function for its runtime [29].

3.3.1.2 Integer Programming Formulation

The basic Traveling Salesman Problem can be modeled by an integer program in which binary variables indicate the decision whether or not to traverse an arc in the network. The variable, x_{ij}, will take on the value of one if the arc between node *i* and node *j* is traveled, and zero otherwise.

The most popular model, proposed by Dantzig, Fulkerson, and Johnson, is given here [9]. The variable d_{ij} is the distance from node *i* to node *j*:

$$\textit{Minimize} \qquad \sum_{(i,j)\in N} d_{ij}x_{ij} \qquad\qquad (3.1)$$

$$\textit{Subject to} \qquad \sum_{i\in N} x_{ij} = 2 \qquad \forall j \in N \qquad (3.2)$$

$$\sum_{i,j\in S} x_{ij} \le |S| - 1 \qquad \forall S \subset N, S \neq \varnothing \, (3.3)$$

$$x_{ij} \in \{0,1\} \qquad\qquad (3.4)$$

The set *N*, as defined above is the set of nodes in the network, while *S* is a subset of *N*. Constraints 3.1 ensures that each node in the network is visited by forcing the traveler to travel into and out of each node. Constraints 3.2 are sub-tour elimination constraints; they ensure that the solution is a single tour. The size of the problem increases dramatically with the addition of the sub-tour elimination constraints; a complete problem includes every sub-tour elimination constraint. This means that there are 2^N sub-tour elimination constraints in the complete problem.

3.3.1.3 Exact Solution Methods

Unfortunately, the Traveling Salesman Problem is difficult to solve, as discussed in Section 3.3.1.1. All known solution methods run in non-polynomial time; in the worst case, they might have to search the entire solution space in order to confirm an optimal solution. However, there are methods to reduce the solve time for the model. These methods attempt to intelligently search the solution space. Three of the most common methods are the *cutting planes method, branch-and-bound,* and *dynamic programming*.

Dantzig, Fulkerson, and Johnson first proposed a *cutting planes method* to solve the integer program. The method begins by solving the *liner programming (LP) relaxation,* which allows the x_{ij} variables to take on continuous values between zero and one. This provides a lower bound for the integer solution. Then, sub-tour elimination constraints are added to force fractional solutions towards integer solutions, as well as to get rid of any sub-tours in the relaxation solution [16].

The *branch-and-bound* method is commonly referred to as a "divide and conquer" algorithm [9]. Similar to the cutting planes method, the first step is to solve the relaxation, and use the solution as a lower bound. The algorithm continues by solving subproblems in an attempt to find integer solutions. Lower bounds are used to discard certain subsets of the feasible set from consideration [29].

Dynamic programming attempts to solve integer problems by sequentially working through the system to find the "best" solution at every point along the way [9]. The idea, first applied to the Traveling Salesman Problem by Richard Bellman, is built upon the concept that the shortest path through a subset of cities must be part of the overall solution [1]. Bellman explained the dynamic programming in the following manner:

> "It is clear that, the tour being the path *i* through $j_i, j_2, \ldots, j_k$ in some order and then to 0 must be of minimum length; for, if not the entire tour could not be optimal since its total length could be reduced by choosing a shorter path from *i* through $j_i, j_2, \ldots, j_k$ to 0" [7].

The algorithm takes advantage of this by finding the shortest route through a subset of the cities. The cities are iteratively added to the route. In each round, starting with the shortest route from the current solution (which contains only a subset of the cities), a new city is added in the way that minimizes the necessary additional distance.

An advantage of dynamic programming is that necessary constraints on the order of the cities visited, which decrease the size of the solution pool, can be incorporated into the algorithm. In addition, the constraints allow the algorithm to dismiss large sets of possible solutions; therefore, with constraints, the algorithm will produce the optimal solution in a shorter amount of time [7].

3.3.1.4 Heuristic Solution Methods

Ideally, heuristic solution methods find near-optimal solutions quickly. These methods do not search the entire set of solutions, but find for the best solution in a reasonable amount of time.

Heuristic methods can be broken down into two groups: first, *tour construction procedures* efficiently build feasible routes by adding vertices one at a time; and, second, *tour improvement procedures* improve an already feasible route [27]. An algorithm that incorporates both tour construction and tour improvement procedures is called a *composite algorithm* [27]. Most algorithms that are used to solve the Traveling Salesman Problem are of this type.

The *nearest neighbor* algorithm is a simple tour construction procedure commonly used to solve the Traveling Salesman Problem [19]. The algorithm starts at an arbitrary starting point, and proceeds to add the node that is closest to the present node until all nodes are included in the path. The last node is then connected to the origin to create a tour.

Another class of construction procedures, known as *insertion algorithms*, follows the following steps [27]:

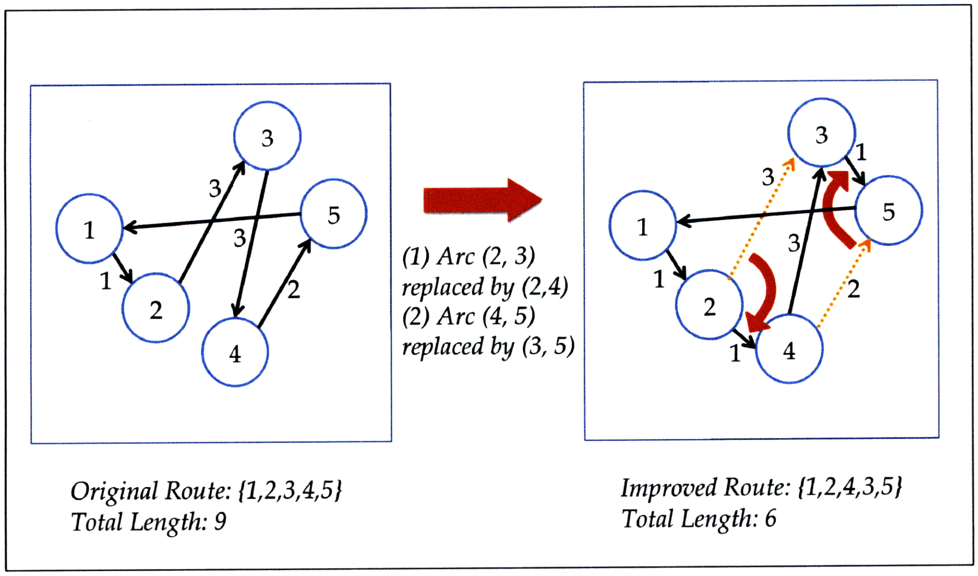
Step 1: Construct a simple tour with only two nodes.

Step 2: Consider each node not in the tour. Insert the node that meets a specific criterion.

Common criterion that are used to decide which node should be added to the network include: (1) Adding the node that is closest to the tour; (2) Adding the node that is the furthest from the tour, known as *furthest neighbor*, and (3) Adding the node that produces the least increase in distance, called *cheapest insertion* [27]. Other criterion have been proposed, including those measure angles and ratios that include multiple metrics [38].

Tour improvement procedures start with a feasible route built by a construction procedure. The procedures then use simple routines to improve the route. Flood noticed that if the path crosses itself at any point during the tour, then the tour could be improved by switching the order of nodes so that the tour does not cross [19]. Croes proposed a similar idea, that he labeled *inversion*, where the order of two nodes is switched to find a better tour [15]. Lin and Kernighan built upon these ideas, proposing the *k-opt algorithm* [1]. The algorithm goes through all subsets of *k* arcs and attempts to reconnect the tour with a set of *k* new arcs. If an improvement can be found, then the new arcs are added to the tour, and the algorithm continues to the next set. An example of the *k*-opt algorithm, the 2-opt algorithm, is shown in Figure 3.7.

Another group of tour improvement procedures are *metaheuristics*. These methods include subroutines that determine which part of the solution set to explore. These heuristics include *simulated annealing*, which is modeled after the cooling process of metal, and *tabu search*, which records which solutions have already been considered [25].



(1) Arc (2, 3) replaced by (2,4)
(2) Arc (4, 5) replaced by (3, 5)

Original Route: {1,2,3,4,5}
Total Length: 9

Improved Route: {1,2,4,3,5}
Total Length: 6

Figure 3.7 Lin-Kernighan 2-opt Algorithm

3.3.2 *Modified Traveling Salesman Problems*

This section describes problems that are similar to the UAV Planner Problem. While fundamental aspects of the UAV Planner Problem are similar to the traveling salesman problem, many aspects of the problem distinguish it from the original Traveling Salesman Problem. This section will introduce two problems that encompass differing elements of the UAV Planner Problem: the *Prize Collecting Traveling Salesman Problem* and the *Traveling Salesman Problem with Time Windows.*

3.3.2.1 The Prize-Collecting Traveling Salesman Problem

The Prize-Collecting Traveling Salesman Problem is described as a Traveling Salesman Problem where each city has an associated "profit" that the salesman will receive by visiting the city. The objective function remains the same as the original Traveling Salesman Problem; the salesman wants to minimize his travel costs. However, an additional constraint is added to ensure that the salesman achieves a certain amount of "prize."

Balas formulated the problem as an integer program [5]. The formulation uses the decision variable x_{ij} to indicated if arc *(i,j)* is traveled by the salesman. The associated costs of travel are denoted by c_{ij}. The decision variable y_i is used to indicate whether or not node *i* is included in the tour. The variables w_i represent the profits associated with node *i*, while w_0 is the amount of prize money that the salesman must achieve. Assuming that there is no penalty if the salesman does not visit a city, the problem is formulated as:

$$\textit{Minimize} \qquad \sum_{(i,j)\in A} c_{ij} x_{ij} \qquad (3.5)$$

$$\textit{Subject to} \qquad \sum_{j\in N-\{1\}} x_{ij} - y_i = 0 \qquad \forall i \in N \qquad (3.6)$$

$$\sum_{i\in N-\{1\}} x_{ij} - y_j = 0 \qquad \forall j \in N \qquad (3.7)$$

$$\sum_{i\in N} w_i x_{ii} \geq w_0 \qquad (3.8)$$

$$y_i \in \{0,1\}, x_{ij} \in \{0,1\}, \forall (i,j) \in A \qquad (3.9)$$

The formulation includes a final constraint that the tour, denoted by *G(x)* must be a cycle [5]. Notice that there are separate decision variables for the nodes and arcs in the network; similar decision variables will be used in the mixed integer programming formulation for the UAV Planner Problem.

Balas proposes an exact method for solving the problem by studying the polytope that defines the set of feasible solutions [18]. Other proposed solution methods include heuristic algorithms with performance guarantees. Awerbach, et al., proposed a heuristic that begins by using an approximation algorithm to find the minimum spanning tree and replicating nodes to create a tour. This approach has been adopted and improved through other studies by Blum, et al. [10], and Arora and Karakostas [18].

3.3.2.2 The Traveling Salesman Problem with Time Windows

The Traveling Salesman Problem with Time Windows constrains the salesman's visit to each town to be within a time window. The salesman must visit after the lower bound of the time window, l_i, and before the upper bound, u_i. The object of the problem remains the same as the Traveling Salesman Problem; to find the shortest tour that connects all of the cities [4].

Baker proposed a nonlinear program to model this problem. The formulation relies on a single decision variable t_i that denotes the time that the salesman visits city i. An additional decision variable, t_{n+1}, represents the time that the salesman returns to the origin. The variables d_{ij} are the shortest times to travel the arcs (i, j):

$$\begin{array}{llll}
\textit{Minimize} & t_{n+1} - t_0 & & (3.10) \\
\textit{Subject to} & t_i - t_1 \geq d_{i1} & i = 2 \rightarrow n & (3.11) \\
 & |t_i - t_j| \geq d_{1j} & i = 3 \rightarrow n, 2 \leq j \leq i & (3.12) \\
 & t_{n+1} - t_i \geq d_{i1} & i = 2 \rightarrow n & (3.13) \\
 & t_i \geq 0 & i = 1 \rightarrow n+1 & (3.14) \\
 & l_i \leq t_i \leq u_i & i = 2 \rightarrow n & (3.15)
\end{array}$$

Baker solves the problem using a branch and bound method that can appropriately handle the nonlinearity and non-differentiability. Gendreau, et al, propose a heuristic approach that constructs tours using a nearest neighbor algorithm [20]. At each iteration, all cities after the city inserted must be checked to ensure that the time window bounds are not violated.

3.3.3 *The Orienteering Problem*

While the previous two formulations presented incorporate aspects of the UAV Planner Problem, the problem closest to the UAV Planner Problem is the *Orienteering Problem* and its derivatives, the *Orienteering Problem with Time Windows* and the *Team Orienteering Problem*. The Orienteering Problem, posed first by Tsiligrides, gets its name from the sport of orienteering, in which competitors travel to pre-determined locations and receive points for each location to which they are able to travel [42]. The competitors have a time limit, within which, they must return to the starting location [23].

The objective of the Orienteering Problem, like the UAV Planner Problem, is to maximize the total value of all locations visited by a competitor. This is a significant deviation from the problems formerly discussed where finding the shortest route was the goal of the problem.

The Orienteering Problem is known to be *NP-complete* [23] and few exact algorithms have been proposed. Ramesh, Yoon, and Karwan developed an exact method using a branch-and-bound algorithm that utilizes the Lagrangian relaxation and problem reformulation to obtain an integer solution [35].

Most research on the Orienteering Problem focuses on heuristic approaches to the problem. The first algorithm proposed by Tsiligrides utilizes a Monte Carlo approach to build initial routes [42]. The routes are then improved through local search heuristics, including methods similar to Lin-Kernigan's 2-opt method.

Further research into the Orienteering Problem continued to produce similar composite algorithms with different methods for building and improving routes. Golden, Levy, and Vohra developed an algorithm that starts with route construction through a cost-benefit analysis [23]. They improve the initial route with a 2-opt method followed by a center-of-gravity improvement method. Golden, Wang, and Liu, produced a more efficient algorithm in which the algorithm "learns" over the course of the improvements [24]. Ramesh and Brown [34] developed an iterative method with four phases: (1) Build initial routes through a cost-benefit analysis scheme; (2) Use Lin-Kernighan 2-opt method to improve routes; (3) Interchange nodes by deleting a node and replacing it with a more valuable node; and (4) Iterating through the phases until the marginal improvement of a round falls beneath a specified threshold [34].

Variants of the Orienteering Problem have also received the attention of researchers due to constraints in the operational problem. The *Orienteering Problem with Time Windows* introduces a range of times for each location in which it must be visited to be of worth to the competitor, similar to the time windows place on observations in the UAV Planner Problem.

Kantor and Rosenwein propose two heuristics to solve the Orienteering Problem with Time Windows [26]. Their first heuristic, called a tree heuristic, "systematically generates a list of feasible paths and then selects the most profitable path from the list." They also develop a composite algorithm in which routes are built through cost-benefit analysis and then improved by inserting nodes where feasible. They found that their tree heuristic found better solutions, but was more computationally difficult. Righini and Salani propose an exact method to solving the Orienteering Problem with time windows using dynamic programming in conjunction with state space relaxation [36].

Both an exact and an approximate algorithm have been proposed for the *Team Orienteering Problem*. In the team Orienteering Problem, a team of competitors can split up to visit locations and their score is the sum of the values of all locations visited by each team member. Therefore, mathematically, the solution would include multiple routes, one for each team member. Boussier, Feillet, and Gendreau use a branch and price algorithm to find an exact solution to the team Orienteering Problem [11]. Vansteenwegen, et al, have developed a metahueristic that uses a greedy construction algorithm to build routes and local search heuristics to improve upon them [43].

3.4 Mixed Integer Programming Model

As with the problems previously presented, a mixed integer program can be formulated to model the UAV Planner Problem. The model provides the ability to explore methods to find the exact solution. Although previous research has shown that exact methods might not be practical, the optimal solution will be useful to evaluate the quality of the solutions generated by heuristic methods.

3.4.1 Mixed Integer Programming Formulation for USV Planner

In the study on the operations of USVs, Miller posed a problem similar to the UAV Planner Problem [30]. The *USV Observation-Planning Problem (USVOPP)* creates operations plans for unmanned surface vessels that collect data from their location on the surface of the ocean. In his study, he develops a mixed integer program for the USV planner problem that is used as the basis for the MIP presented for the USVOPP. Alterations, such as the inclusion of multiple locations for each task, were added to the formulation.

The MIP developed by Miller took advantage of integer decision variables that include the placement of a node in the route. The variable x_{ikt} takes a value 1 if node i is visited by USV k in the t-th placement on the route. The variable, therefore, models multiple decisions. A similar variable is used in the MIP formulation for the UAV Planner Problem; in this formulation, the variable includes the location at which a task should be performed.

Continuous decision variables are used to model the time at which nodes are visited. Similar to the formulation of the Traveling Salesman Problem with Time Windows given in Section 3.3.2.2, there is a lower bound and upper bound within which the task must be completed.

3.4.2 Model Formulation

This section introduces the mixed integer programming model for the UAV Planner Problem. First, the sets, decision variables, and inputs notation are defined. Then, the objective function and constraints are introduced.

3.4.2.1 Set Definitions

The following sets will be used in the formulation:

C = Set of task, location pairs
T = Set of all tasks
U = Set of UAVs
P = Set of placements in a path

The set P is the set of placements in a path. A *placement* denotes the order of a task in the path. For example, if a task is in placement five in a path, then the task is the fifth task that will be performed by the UAV. The set of placement contains the placement for each task in the path; therefore, the set associated with a path with ten tasks will contain the first ten natural numbers while the set of placements associated with a path containing only five tasks will contain the first five natural numbers.

3.4.2.2 Decision Variables

$perform_{(i,k),(u,t)}$ A binary decision variable describing if task i is performed at location k by UAV u in placement t. The t describes the location of the task in the respective UAV's path. For example, if t=1, then task i at location k will be the first task completed by UAV u.

$travel_{(i,k),(j,l),u}$ A binary decision variable describing if the arc from *(i,j)* to *(j,l)* is traveled by UAV *u*.

$arrive_{(i,k),u}$ A continuous variable that assigns the time that UAV *u* will arrive at task *i* in location *k*.

$depart_{(i,k),u}$ A continuous variable that assigns the time that UAV *u* will depart task *i* in location *k*.

3.4.2.3 Input Notation

The inputs to the model are those discussed in Section 3.1.2. They are denoted as:

$early_i$	Beginning of time window for task *i*
$late_i$	End of time window for task *i*
$obstime_i$	Required time to complete task *i*
$altitude_{(i,k)}$	The altitude of task *i* at location *k*
$horizon$	Planning horizon
$ceiling_u$	Maximum altitude for UAV *u*
$floor_u$	Minimum altitude for UAV *u*
$endurance_u$	Maximum length of flight for UAV *u*
$traveltime_{(i,k),(j,l),u}$	Length of time for UAV *u* to travel from location *(i,j)* to *(j,l)*
$value_{(i,k),u}$	The value for UAV *u* to complete task *i* at location *k*

3.4.2.3 Objective Function

The objective is to maximize the total value of tasks completed. Mathematically, this is:

$$Maximize \qquad \sum_{u \in U} \left(\sum_{(i,k) \in C, t \in P} value_{(i,k),u} * perform_{(i,k),u,t} \right)$$

3.4.2.5 Constraints

The model has thirteen constraints that are categorized as either UAV operational constraints, network constraints, or time window constraints. The UAV constraints ensure that the capabilities of the UAVs are not exceeded, ensuring that the resulting operations plan is feasible for the vehicles. The following are the UAV operational constraints:

(1) Ensure that the observation altitude is above the UAV floor

$$altitude_{(i,k)} * \sum_{u \in U} \sum_{t \in P} perform_{(i,k),u,t} \geq floor_u \qquad \forall (i,k) \in C, u \in U$$

(2) Ensure observation altitude is below UAV ceiling

$$altitude_{(i,k)} * \sum_{u \in U} \sum_{t \in P} perform_{(i,k),u,t} \leq ceiling_u \qquad \forall (i,k) \in C, u \in U$$

(3) Ensure all activities are completed within the planning horizon

$$arrive_{(i,k),u} + obstime_i \leq horizon \qquad \forall (i,k) \in C, u \in U$$

The network constraints ensure that the resulting operations plan creates a feasible path plan for the operations plan. The following are the network constraints:

(4) Each placement, *t*, can only be assigned one task

$$\sum_{(i,k) \in C} perform_{(i,k),u,t} \leq 1 \qquad \forall u \in U, t \in P$$

(5) Constrain a single UAV to perform a task

$$\sum_{u \in U} \sum_{k \in L} perform_{(i,k),u,t} \leq 1 \qquad \forall i \in T, t \in P$$

(6) Ensure that the targets are assigned in successive placements on the UAV path

$$\sum_{(i,k) \in C} perform_{(i,k),u,t+1} - \sum_{(i,k) \in C} perform_{(i,k),u,t} \leq 0 \qquad \forall u \in U, t \in P - 1$$

(7) Force an arc to exist between two successive performed tasks

$$perform_{(i,k),(u,t+1)} + perform_{(i,k),(u,t)} - travel_{(i,k),(j,k),u} \le 0 \qquad \forall (i,k) \in C,(j,l) \in C,u \in U,t \in P-1$$

(8) Force an arc to leave every performed node

$$\sum_{u \in U} travel_{(i,k),(j,l)u} - \sum_{u \in U}\sum_{t \in P} perform_{(i,k),(u,t)} \le 0 \quad \forall (i,k) \in C,(j,l) \in C$$

(9) Ensure than arc enters every performed node

$$\sum_{u \in U} travel_{(i,k),(j,l)u} - \sum_{u \in U}\sum_{t \in P} perform_{(i,k),(u,t)} \le 0 \quad \forall (i,k) \in C,(j,l) \in C$$

The time window constraints limit the resulting operations plan to performing tasks within the desired time window. In this formulation, the observation must be completed within the time window. The following are the time window constraints:

(10) UAV must arrive after the beginning of the time window

$$arrive_{(i,k),u} \ge early_i * perform_{(i,k),u,t} \qquad \forall (i,k) \in C,u \in U,t \in P$$

(11) UAV must exit before end of time window

$$depart_{(i,k),u} \le late_i * perform_{(i,k),u,t} \qquad \forall (i,k) \in C,u \in U,t \in P$$

(12) UAV must depart the task after sufficient time to complete the task

$$depart_{(i,k),u} \ge arrive_{(i,k),u} + obstime_i * perform_{(i,k),u,t} \quad \forall (i,k) \in C,u \in U$$

(13) Ensure sufficient travel time between tasks ($a_{(i,k),(j,l),u}$ must be calculated a priori as explained in Subsection 3.4.2.6)

$$depart_{(i,k),u} + traveltime_{(i,k),(j,l),u} - a_{(i,k)(j,l),u}(1 - travel_{(i,l),(j,l),u}) \leq arrive_{(j,l),u}$$

$$\forall (i,k) \in C, (j,l) \in C$$

3.4.2.6 Linearization of Constraint (13)

The value for $a_{(i,k),(j,l),u}$ in constraint (13) is a linearization of a necessary constraint. The model needs to constrain the time that a UAV arrives at a next task to be greater than the time that it departs the previous task plus the travel time between the two. The natural way to write this constraint would be:

$$\left(depart_{(i,k),u} + traveltime_{(i,k),(j,l),u}\right) * \left\| \left[\left(perform_{(i,k),u,t} - perform_{(j,l),u,t+1}\right) - 1 \right] \right\| \leq arrive_{(j,l),u}$$

However, this constraint is not linear, because variables are being multiplied together and an absolute value is used. Therefore, Ropke, Cordeau, and Laporte, developed the following way to linearize the constraint [37]. The value, $a_{(i,k),(j,l),u}$, which must be found before solving the model, is:

$$a_{(i,k)(j,l),u} = \max(0, late_i + obstime_i + traveltime_{(i,k)(j,l),u} - early_j)$$

This constraint will now either be redundant with the non-negativity constraints (if the UAV does not travel between the two locations) or will constraint the arrival to the next task to be greater than the departure time of the last task plus the travel time.

3.4.4 Implementation

Optimization software can be used to solve the MIP. The software chosen is ILOG's OPL Studio 5.5 which utilizes CPLEX 11.0. This software utilizes solution methods discussed in the Literature Review; the method is based on the branch-and-bound method. The software provides an exact solution to the mixed integer program, and thus an optimal solution to the UAV Planner Problem. The results and performance of this implementation will be discussed in detail in Chapter 5.

Chapter 4

Formulation of Algorithm

When solved, the mixed integer program introduced at the end of the previous chapter provides an exact solution to the UAV Planner Problem. While the MIP is beneficial in solving the UAV Planner Problem, the complexity of the MIP might cause lengthy runtimes. This motivates the development of an algorithm to generate operations plans in a reasonable amount of time.

This chapter introduces an algorithm, called the *Composite Operations Planning Algorithm (COPA).* COPA combines a metaheuristic with a linear program to solve the UAV Planner Problem. The description of COPA begins with an introduction to *composite variables.* A composite variable is a binary decision variable that models multiple related decisions. The UAV Planner Problem is reformulated using composite variables that represent a path and an associated type of UAV. The reformulation is incorporated into an algorithm that uses a metaheuristic to generate the composite variables.

The chapter begins by describing composite variables and introducing the reformulation of the UAV Planner Problem. The chapter continues by describing COPA and the steps of the algorithm. The end of the chapter discusses the implementation of COPA, known as the COPA software.

4.1 Composite Variable Formulation

A *composite variable* is a decision variable that models multiple decisions in a single variable. The decisions are not necessarily connected; in fact, the variables often group together decisions that do not initially seem to be related.

For the UAV Planner Problem, composite variables can simplify the formulation of the problem. Many of the complexities of the problem, such as time windows and three-dimensional paths, can be incorporated into a composite. Therefore, these aspects do not have to be explicitly modeled; decreasing the number of variables and simplifying the formulation. The following provides an example of a composite variable.

Example 4-1. *In this example, a composite combines a path on a graph, G, and a vehicle type. The graph, G, has four nodes, {1, 2, 3, 4, 5}, and two types vehicles, {A, B}, travel on the graph. The following are paths through the graph G:*

Path (1): 1 → 2 → 5

Path (2): 1→ 3 → 4 → 5

Path (3): 1 → 4 → 5

The paths are shown in Figure 4-1. The composite variable y_i^k *represents the decision for a vehicle of type* k *to travel along Path i. Therefore, if* y_1^A *takes the value 1, then a vehicle of type A travels on Path (1); if* y_1^A *takes the value 0, then a vehicle of type A does not travel on Path (1). The variable includes two decisions that are generally not connected: first, whether Path (1) is traveled on; second, the specific type of vehicle to travel on the path.*

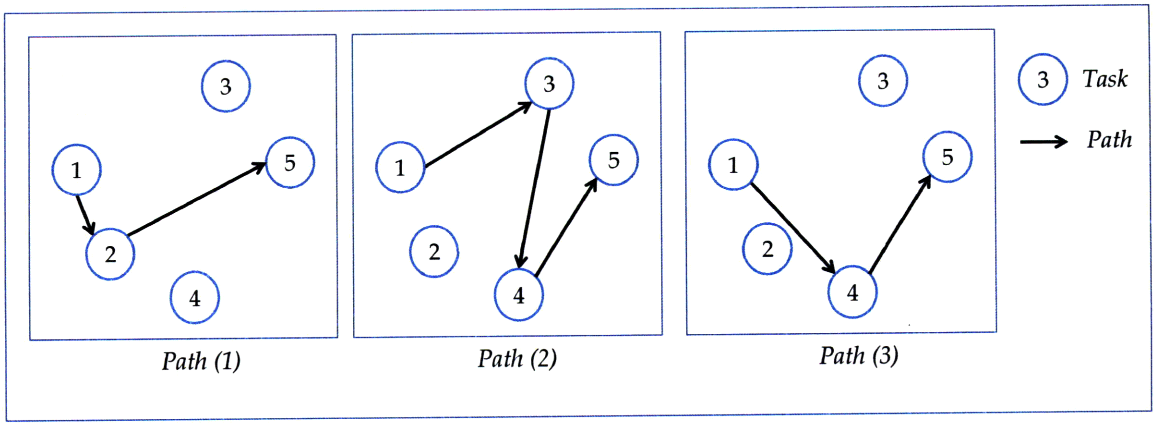


Figure 4.1 Paths for Example 4-1

4.1.1 Literature Review of Composite Variables

This section reviews some of the literature on composite variables. Much of the literature on composites applies the technique to transportation problems, incorporating path and flow variables into composites. This section focuses on transportation related problems, although the approach has been modified for other types of problems, including general network design and fixed-charge problems [33].

Armacost, Barnhart, and Ware use composite variables to design the air network for the United Parcel Service [3]. Their formulation for the design problem models aircraft routes and package flow as composite variables. The solution to their composite variable formulation provides an air network for the carrier's next day of operations. It is shown that the composite variable formulation achieves a stronger lower bound than traditional network design approaches. Focusing on the same problem, Armacost presents the composite variable formulation as a Dantzig-Wolfe decomposition of a traditional network design formulation [2]. In addition, he compares the composite variables to Chvátal-Gomory cuts in the dual of the formulation.

Using a similar definition for composite variables, Nielsen addresses a network design problem to transport military personnel and cargo [33]. Nielsen's formulation combines one or more aircraft missions that cover the complete movement of a subset of cargo into a composite variable. He concludes that the formulation has strong linear programming relaxations and can design a large-scale transportation network in a short solution time.

Cohn and Barnhart apply a composite variable formulation to crew scheduling and maintenance routing decisions for commercial airlines [13]. The formulation combines the two problems—maintenance routing and crew scheduling—which are typically solved sequentially. Solving the problems separately can result in inefficiencies in the system, which is costly for the airlines. Cohn and Barnhart use composite variables to integrate the problems and provide more efficient plans for the airlines [13].

Barth provides a military operations planning application using composite variables [6]. He focuses on the real-time planning of aircraft missions, including re-planning during the execution of a mission. The formulation uses composite variables that include targets, multiple aircraft, and routes. He concludes that the formulation has a strong linear programming relaxation when compared to traditional methods. In addition, the formulation can be solved quickly to allow dynamic re-planning of the missions.

4.1.2 *Full Path Composite Variables*

To create a linear program for the UAV Planner Problem using composite variables, the composite variables should be designed to incorporate multiple decisions that must be made in creating operations plans for the UAVs. A *full path composite,* presented in this section, is designed to model necessary decisions in the planning process.

A full path composite has two parts: a full path and a type of UAV. The full path is an ordered set of tasks that spans the planning horizon. Therefore, a UAV can only perform one full path per planning horizon. The full path composite also has an assigned type of UAV. If the composite is incorporated into the operations plan, then a UAV of the type will be assigned to fly the path. A *full path composite variable* is a binary variable that represents the decision whether or not to incorporate the full path composite associated with the variable into the operations plan.

A feasible set of full path composite variables provides the information needed to create an operations plan. Example 4-2 illustrates a full path composite variable.

Example 4-2. *A full path composite variable includes a full path with an assigned UAV type. The full path composite variables are similar to the composite variables introduced in Example 4-1; however, this example incorporates multiple locations per task into the composite variables.*

Consider a scenario with five tasks denoted {1, 2, 3, 4, 5} that can be performed at two locations each. The UAVs start at a command station, denoted by 0. The notation for this is (i, j) where i denotes the task number and j denotes the location number. There are two vehicle types denoted {A, B}. The following are full path composite variables:

Composite (1): UAV type A completes the path 0 → (1, 1) → (4, 1) → (2, 1) → 0

Composite (2): UAV type B completes the path 0 → (2, 1) → (5, 2) → (1, 1) → 0

Composite (3): UAV type A completes the path 0 → (4, 2) → (3, 1) → (1, 1) → 0

The composites are shown in Figure 4.2. Similar to the previous example, each composite can be represented by a single variable. In this case, the binary variable y_1 represents the decision whether or not to include Composite (1) in the operations plan. The variable y_1 takes a value of 1 if Composite (1) is included in the operations plan; otherwise, y_1 takes a value of 0.

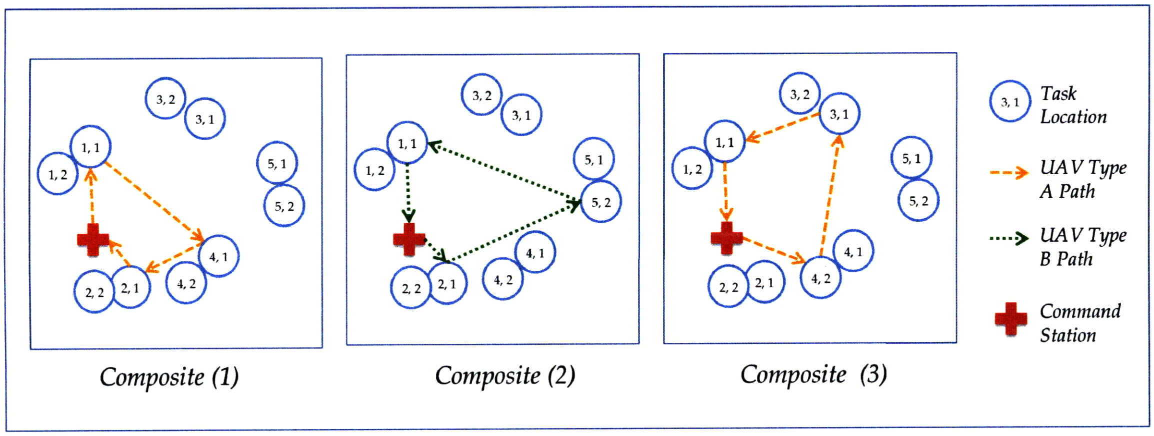


Figure 4.2 Composites for Example 4-2

Example 4-2 illustrates that full path composite variables model multiple decisions; for example, the decision of the location that a task should be performed at is incorporated into the path. When a composite variable is selected, each of the following decisions is made: the time period in which the tasks are performed, the location at which tasks are performed, and the type of UAV that performs the task.

4.1.3 Composite Variable Formulation

The composite variable formulation utilizes the full path composite variables introduced in the previous section. This formulation models the UAV Planner Problem and a solution contains the information needed for an operations plan.

The full path composite variables are used in a binary program for the UAV Planner Problem. The following sets are used in the formulation:

C = *Set of Composite Variables*

T = *Set of Tasks*

A = *Set of UAV Types*

U_a = *Set of UAVs of Type a*

The formulation has a single binary decision variable that describes if composite *c* is included in the operations plan:

$$\gamma_c = \begin{matrix} 1, \textit{ if composite } c \textit{ is included} \\ 0, \textit{ otherwise} \end{matrix}$$

Inputs for the formulation include information about the composites:

$$\delta_c^t = \begin{cases} 1, \textit{ if composite c includes task t} \\ 0, \textit{ otherwise} \end{cases}$$

$$\alpha_c^a = \begin{cases} 1, \textit{ if composite c uses UAV type a} \\ 0, \textit{ otherwise} \end{cases}$$

$$v_c = \textit{The value of composite c}$$

The value of composite *c*, v_c, is found by summing the values of all tasks performed in the path included in composite *c*. With these inputs, the formulation for the *Full Path Composite Variable Binary Formulation* (FPCVBF) is:

$$\begin{array}{llllr} \textit{FPCVBF:} & \textit{Maximize} & \sum_{c \in C} v_c \gamma_c & & (4.1) \\ & \textit{Subject to} & \sum_{c \in C} \delta_c^t \gamma_c \leq 1 & \forall t \in T & (4.2) \\ & & \sum_{c \in C} \alpha_c^a \gamma_c \leq |U_a| & \forall a \in A & (4.3) \\ & & \gamma_c \in \{0,1\} & \forall c \in C & (4.4) \end{array}$$

Constraints 4.2 ensure that each task is performed a single time. Constraints 4.3 allow for a composite variable to be selected for each UAV in the system; the constraints ensure the number of selected composites for a UAV type is less than the number of available UAVs of the type

This FPCVBF greatly simplifies the MIP presented at the end of Chapter 5. However, it maintains the constraints of the UAV Planner Problem. Many of the constraints are incorporated into the generation of the composites; thus, the constraints do not need to be modeled explicitly. For example, when the composite variable path composite is created, it is ensures that the data is collected within time window constraints. Therefore, there is no need for variables to model the start and end times of observations in the composite formulation. This reduces the number of variables and constraints necessary to model the problem.

4.1.4 *Integrality of Linear Relaxation*

The formulation given above is a binary program, because the decision variable γ_c is constrained to be in the set {0,1} in Constraints 4.4. However, the constraint can be relaxed and it will still result in a binary solution. This is due to the structure of the problem; specifically, the structure of the constraints. The constraints create a feasible region with vertices where the values of the variables are either zero or one. Because of this, the relaxed version of the formulation provides an answer that is at the vertices of the feasible region.

The problem can be reformulated with a relaxed constraint for the composite variables. The reformulation is called the *Full Path Composite Variable Formulation*:

$$\begin{array}{llllr}
\textit{FPCVF:} & \textit{Maximize} & \sum_{c\in C} v_c\gamma_c & & \textit{(4.5)} \\
 & \textit{Subject to} & \sum_{c\in C} \delta_c^t\gamma_c \le 1 & \forall t\in T & \textit{(4.6)} \\
 & & \sum_{c\in C} \alpha_c^a\gamma_c \le |U_a| & \forall a\in A & \textit{(4.7)} \\
 & & 0\le \gamma_c \le 1 & \forall c\in C & \textit{(4.8)}
\end{array}$$

4.1.5 Using a Metaheuristic for Full Path Composite Variables

A *metaheuristic* utilizes other methods in conjunction with a heuristic to solve a problem. For the FPCVF, a metaheuristic can be used to generate the composite variables for the formulation.

As discussed in Section 3.3.1, heuristics are commonly used to solve problems similar to the UAV Planner Problem. The heuristic methods discussed in 3.3.1 can be used to generate full path composites. The full path composites are represented by composite variables in the FPCVF. In this way, the heuristic that generates the composites is a subroutine that is used in conjunction with linear programming to solve the UAV Planner Problem, as described in Section 4.2.

4.2 Composite Operations Planning Algorithm (COPA)

This section presents the *Composite Operations Planning Algorithm (COPA)* that is developed to solve the UAV Planner Problem and uses FPCVF as its optimization model. Section 4.2.1 provides an overview of the algorithm. Sections 4.2.2 through 4.2.4 describe the steps of the algorithm in detail.

4.2.1 Methodology

COPA combines the use of a linear program and a metaheuristic to solve the UAV Planner Problem. The algorithm starts by iterating through the metaheuristic multiple times until a set number of composites have been generated. Once the metaheuristic is completed, the composites are used in the FPCVF to solve for the optimized set of path plans for the entire set of UAVs.

COPA includes three main steps. The steps, which are discussed in more detail in Sections 4.2.2 through 4.2.4, are introduced here:

> ***Step I: Subset Allocation.*** *In this step, tasks are allocated to be performed by a specific UAV type.*
>
> ***Step II: Composite Generation.*** *Using the set of tasks allocated to a UAV type, full path composites are generated for each UAV of the type. This step utilizes a heuristic method to create the path plans.*
>
> ***Step III: Full Path Composite Variable Linear Program.*** *The full path composites generated in Step II are modeled by full path composite variables in the FPCVF. The FPCVF is solved to determine the best set of composites to incorporate into the operations plan for the system of UAVs.*

Steps I and II are iterated until a predetermined number of composite are generated for the FPCVF; this number might be limited by computer memory or runtime constraints. Step III is solved a single time and the solution is used to create an operations plan for the entire set of UAVs. Figure 4.3 illustrates the flow of the algorithm.

The role of the FPCVF in the algorithm has two purposes. First, it ensures that the best set of generated path plans are chosen to create the observation plan for the system. Second, it ensures that the solution is a *set cover*: that each UAV in the system is assigned a path plan so that the solution utilizes each UAV.

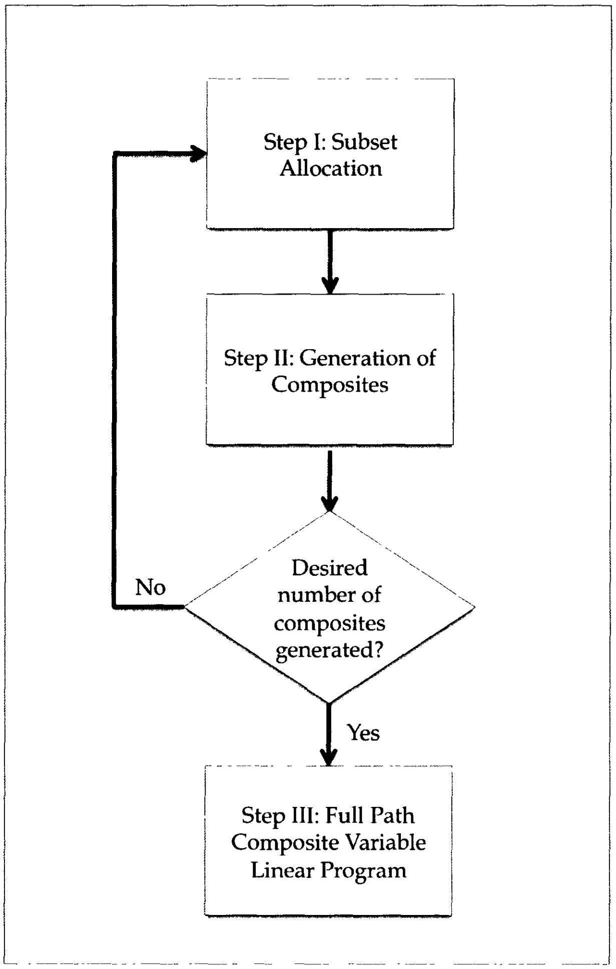

Step I: Subset Allocation
Step II: Generation of Composites
Desired number of composites generated?
No
Yes
Step III: Full Path Composite Variable Linear Program

Figure 4.3 Flow of the Composite Operations Planning Algorithm

4.2.2 *Step I: Subset Allocation*

In the first step of the algorithm, tasks are assigned to be performed by a specific type of UAV. The result is a mutually exclusive, collectively exhaustive set of task subsets, i.e., a partition of the full task set. Each subset has an assigned type of UAV to perform the tasks in the subset.

The objective of this step is to assign tasks to subsets that result in high value composite variable path plans. To achieve this, the assignment of tasks to subsets occurs multiple times and each time a different heuristic method is used to determine which subset to assign a task to.

Each method starts with the same steps. First, it is determined which types of UAVs are able to perform the tasks. For example, for a task location that is far away, a UAV type with a short endurance cannot reach the location. Therefore, this task should not be included in the subset assigned to the UAV with short endurance. Second, the UAV Planner operator can assign tasks to subsets. Each method takes into account these assignments before starting the breakdown of the remaining tasks.

Sections 4.2.2.1 through 4.2.2.3 present methods used for subset allocation. These methods provide simple routines to allocate tasks to the subsets.

4.2.2.1 Observation Length Subsets

The first heuristic method used for subset allocation utilizes the required observation time of the task to determine which type of UAV should perform the task. The supposition is that UAVs with longer endurance are better suited to perform the tasks that have longer observation times. Essentially, the tasks with longer observation times are assigned to the subsets associated with the UAVs with longer endurances.

To describe the method (and following methods) the following notation is used:

A = *Set of UAV types*
U_i = *Set of UAVs of type i*
S_i = *Set of tasks assigned to UAV type i*
T = *Set of Tasks*
u_i = *Number UAVs of type i*
$\boldsymbol{u}$ = *Total number of UAVs*
$\boldsymbol{a}$ = *Total number of UAV types*
i = *Index for type of UAV*
t = *Index for set of tasks*

The following steps describe the method:

(1) Sort the set of tasks, *T*, by observation length. The index used for the set of tasks is *t*; $t = 0$.

(2) Select the UAV type with the longest endurance for which a subset has not been created; this is type *i*.

(3) Assign tasks *t* through $t + u_k/\boldsymbol{u}$ to S_i.

(4) Repeat Steps 2 and 3 until all tasks have been assigned to a subset.

The resulting subsets are then used in Step II of the COPA. The next subsection describes an alternate method for subset allocation.

4.2.2.2 Time Window Subsets

This method for subset allocation utilizes the time windows of each task to determine which subset to assign the task to. The logic behind this method is that each UAV should have tasks whose time windows are spread across the planning horizon. This addresses the problem of UAV types being assigned tasks that are clustered in a portion of the planning horizon, thus unnecessarily limiting the number of tasks that can be completed.

The following steps describe the method. The notation introduced in the previous subsection is used.

(1) Sort T by early time window.

(2) Loop through the following steps until $t = |T|$. Initialize $t = 1, i = 1$:

  (a) Assign task t to S_i.

  (b) Update $t = t + 1; i = i + 1$ if $i < a$, $i = 1$ if $i = a$.

The subsets created by this algorithm are then used in composite generation in Step II of the algorithm. The next subsection describes a third method for subset allocation.

4.2.2.3 Random Subsets

This method for creating subsets uses random numbers to assign the tasks to subsets. Depending on the value of the random number, N, the task being assessed is assigned to any of the possible UAV types, as long as it is feasible for the UAV type to perform the task. The size of the subsets is also randomly generated.

The method uses the same notation as the previous sections. To create the subsets, the following steps are taken:

(1) Initialize $t = 1$.

(2) Loop through the following steps until $t = |T|$:

  a. Get a random number, $N \in [0, 1]$.

  b. Assign task t to a subset using the following rules:

    i. If $N < 1/\boldsymbol{a}$, assign to S_1.

ii. If $(i - 1)/a < N < i/a$, assign to S_i.

iii. If $N > (a - 1)/a$, assign to S_a.

These subsets are also used in Part II of the algorithm. The next section describes how they are used to generate composites.

4.2.3 *Step II: Composite Generation*

In next step of COPA, full path composites are generated for each UAV in the system. A metaheuristic is used to generate the composites. This metaheuristic is based on the heuristic developed by John Miller in his study of the *USVOPP*. The heuristic developed by Miller is called the *3PAA*, or the *Three Phase Approximate Algorithm* [30]. Alterations to Miller's algorithm were made to include multiple locations for each task, which was not a part of the *USVOPP*.

The 3PAA proposed by Miller has the following three stages: the *Construction Phase*, the *Improvement Phase*, and the *Insertion Phase*. First, the Construction Phase uses a construction heuristic that uses cost-benefit analysis to determine which task locations to add to the path plans. After the initial paths are constructed, the algorithm moves into the second phase, the Improvement Phase. In this phase, the initial paths are improved upon using three improvement procedures: *2-Opt*, *Deletion-Insertion*, and *2-Exchange*. In the Insertion Phase, unvisited tasks are inserted into the paths wherever it is feasible. The resulting paths are a solution to the USV Operations-Planning Problem [30].

For the UAV Planner Problem, the 3PAA was modified to include the consideration of different locations at which the tasks can be completed. To accomplish this, task locations are considered in the Construction Phase of the algorithm. In addition, a fourth phase is added after the Insertion Phase, in which the task location can be swapped with a location of higher value for the same task; this additional phase is called the *Location Swap Phase*.

The metaheuristic developed for COPA is described in detail in the following subsections. The metaheuristic has four phases: the Construction Phase, the Improvement Phase, the Insertion Phase, and the Location Swap Phase.

4.2.3.1 Construction Phase

In the construction phase of the metaheuristic, initial paths are created for the UAVs in the system. To create the paths, the construction algorithm builds a path for a UAV of a given type by calculating a cost-benefit ratio and adding the task with the highest ratio to the end of the path.

The construction phase is dependent on input describing the UAV types, the tasks, and the task locations. The input is represented by the notation introduced in Section 4.2.2.1 for the MIP. The notation and additional notation is defined here:

$value_{t,l}$	*Value of task t at location l*
$obstime_t$	*Required time to complete task t*
$idletime_{(t,l)}$	*Length of time that the UAV will have to wait before starting task at location l due to a time window constraint*
$late_t$	*End of time window for task t*
$traveltime_{(i,j)(t,l),i}$	*Length of time for a UAV type i to travel from task location (i, j) to (t, l)*
$endurance_i$	*Endurance of UAV type i*
$horizon$	*Planning horizon*

To ensure time window constraints are kept, the current time in the path must be recorded; it is denoted by *currentTime*. In addition, the parameters w_1, w_2, w_3, and w_4 are used to denote weights used in the cost-benefit ratio.

The following steps describe the construction procedure. The procedure is repeated for each UAV type, and produces the set $P_{i,k}$, the set of ordered tasks for the path of UAV k of type i:

(1) Obtain list of locations associated with tasks in S_i. This is set L_i.

(2) Set $currentTime_i = 0$; current location is (t', l').

(3) For each UAV of type *i*:

   a. For each feasible tasks in L_i calculate:

$$\frac{value_{(t,l)}}{w1 * obstime_t + w2 * idletime_{(t,l)} + w3 * late_t + w4 * traveltime_{(t',l')(t,l),u}}$$

   b. Find $max\{\rho_{t,l}\} \forall (t,l) \in L$; this is (t^*, l^*)

   c. Add (t^*, l^*) to $P_{i,k}$.

   d. Remove all other locations for task t from L_i.

e. Update *$currentTime_k$*. If *$currentTime_k$* > *$endurance_i$* or *$currentTime_k$* > *horizon*, start path for next UAV; $k = k + 1$.

When completed, this procedure provides an initial path, in the form of an ordered set of locations, for each UAV. As previously noted, the ordered set of locations for UAV *k* of type *i* is represented by $P_{i,k}$. The symbol P_i represents the set of paths for UAVs of type *i*; therefore, $P_i = \{P_{i,1}, P_{i,2}, \ldots\}$. The symbol *P* denotes the entire set of paths, which is also the set of composites. The next phase of composite generation improves upon the paths in set *P*.

4.2.3.2 Improvement Phase

The improvement phase of composite generation improves the paths created by the construction phase. To improve the paths, this phase uses three improvement procedures: *2-Opt*, *Deletion-Insertion*, and *2-Exchange*.

Each of the improvement procedures utilizes the order of the tasks in the current path. The *placement* of a task refers to the order of the task in the set. For example, if the current path is represented by the ordered set *{3, 4, 2, 1}* then task 2 is in the third placement in the path.

The procedures use placement to determine which tasks to consider when improving the paths. For COPA, this is generally within 25 percent of the current placement of the task. The percentage is of the total number of tasks in the path; for the example above, 25 percent would be one because there are four tasks. Therefore, in this example, a task is within 25 percent of another task if it is within one placement of the task.

4.2.3.2.1 2-Opt

The first improvement procedure is modified from the Lin-Kernigan 2-opt method [30]. In a 2-opt, two arcs on the path are replaced by two new arcs. If the change shortens the duration of the path and the path remains feasible, then the new arcs are incorporated into the path. This also changes the order of the tasks in the path.

The time window aspect of the UAV Planner Problem causes many paths to be infeasible after a 2-opt is performed. Therefore, the traditional 2-opt can be modified to increase the likeliness that the resulting path remains feasible. For this purpose, tasks are limited to being swapped with tasks that are near each other in the path; this is defined to be within 25 percent of the total number of tasks in the path. For example, if the path includes twelve tasks, then the tasks must within four placements of each other in the existing path. Example 4-3 illustrates the 2-opt algorithm.

Example 4-3. *In this example, the existing path for a particular UAV type is represented by the ordered set of tasks {a, b, c, d, e, f, g, h}.*

*When considering swapping Task c with another task, only tasks within two placements of Task c are considered because there are eight tasks in the path (8*0.25 = 2). Therefore, the following tasks are eligible to be swapped with task c: a, b, d, and e. Let's consider switching the order of Task b and Task c. Arcs (a, b) and (c, d) are replaced by arcs (a, c) and (b, d). The feasibility of the new path is checked (including the time windows for the tasks). If the path is still feasible, then the new duration is calculated and compared to the duration of the original path. If the new duration is less, then the new path is kept. An illustration of this example is show in Figure 4.4.*

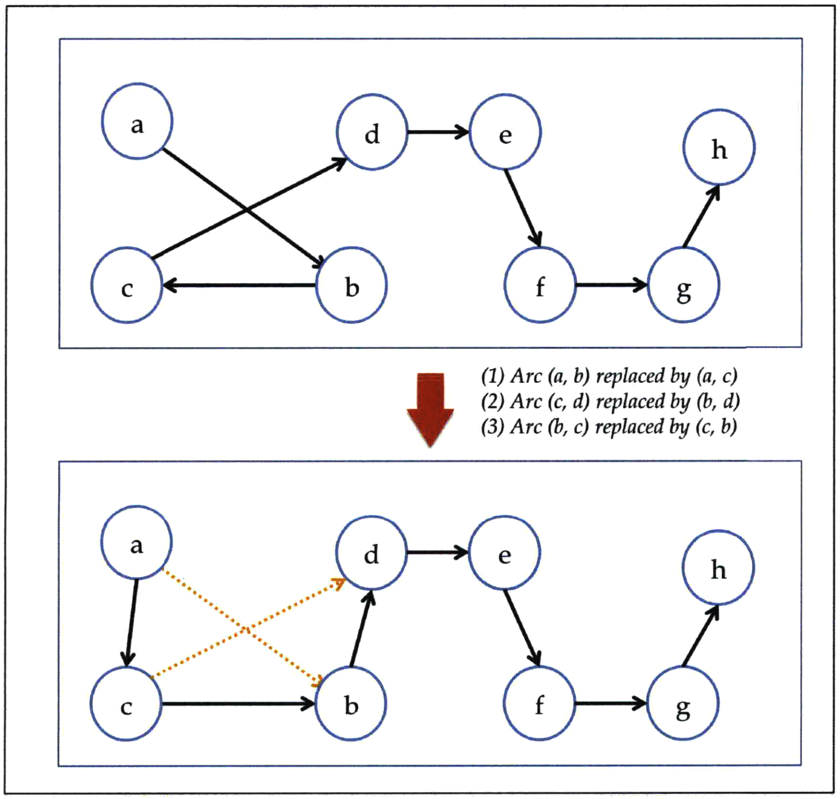

a
d
e
h
c
b
f
g
(1) Arc (a, b) replaced by (a, c)
(2) Arc (c, d) replaced by (b, d)
(3) Arc (b, c) replaced by (c, b)
a
d
e
h
c
b
f
g

Figure 4.4 2-opt Example

4.2.3.2.2 Deletion-Insertion

The Deletion-Insertion method is an *inter-path* method; the method deals with two separate paths when attempting to improve the solution. The Deletion-Insertion method deletes a task from the first path, and then inserts the task into another path.

As with the 2-opt method, the Deletion-Insertion method can cause infeasibility in the time window constraints on the tasks. The method is altered similarly to the 2-opt method. The method considers inserting tasks that are proportionally in the same placement on the new path as the task was in the old path. For example, if the old path has twelve tasks and the fourth task is being considered for deletion, then it is calculated that the task is currently located at the 25th percentile in order of the path. If the new path has eight tasks, then proportionally, that would be the second task, so inserting it in the second placement in the path would be considered.

Insertion is considered within 25 percent of the proportional placement. Because the new path has eight tasks, 25 percent is within two placements of the proportional placement. Therefore, in this example, the task can be inserted in placement 1, 2, 3, or 4 of the new path. The deletion and insertion are maintained if the resulting paths have a cumulative shorter duration and if they remain feasible. Example 4-4 demonstrates the Deletion-Insertion method.

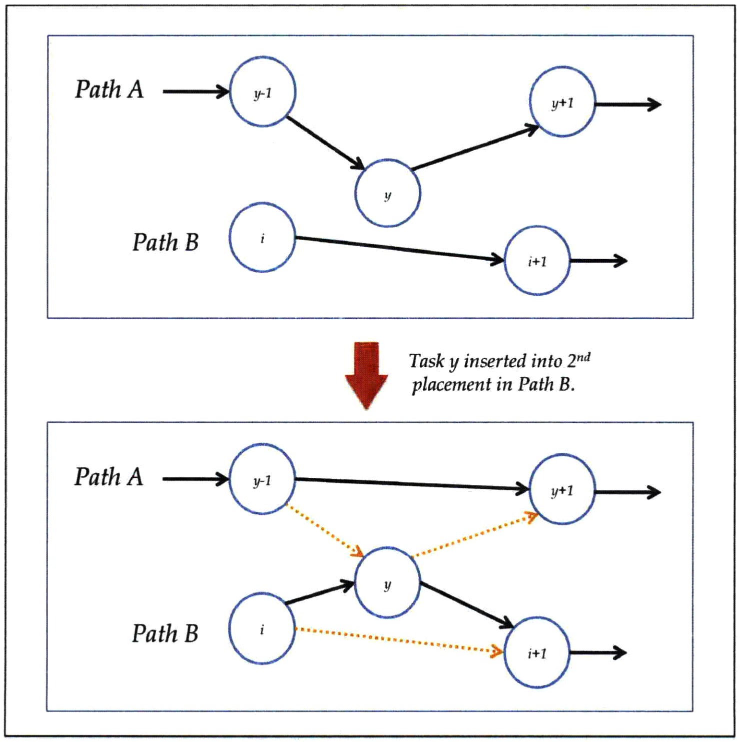


Figure 4.5 Deletion-Insertion Example

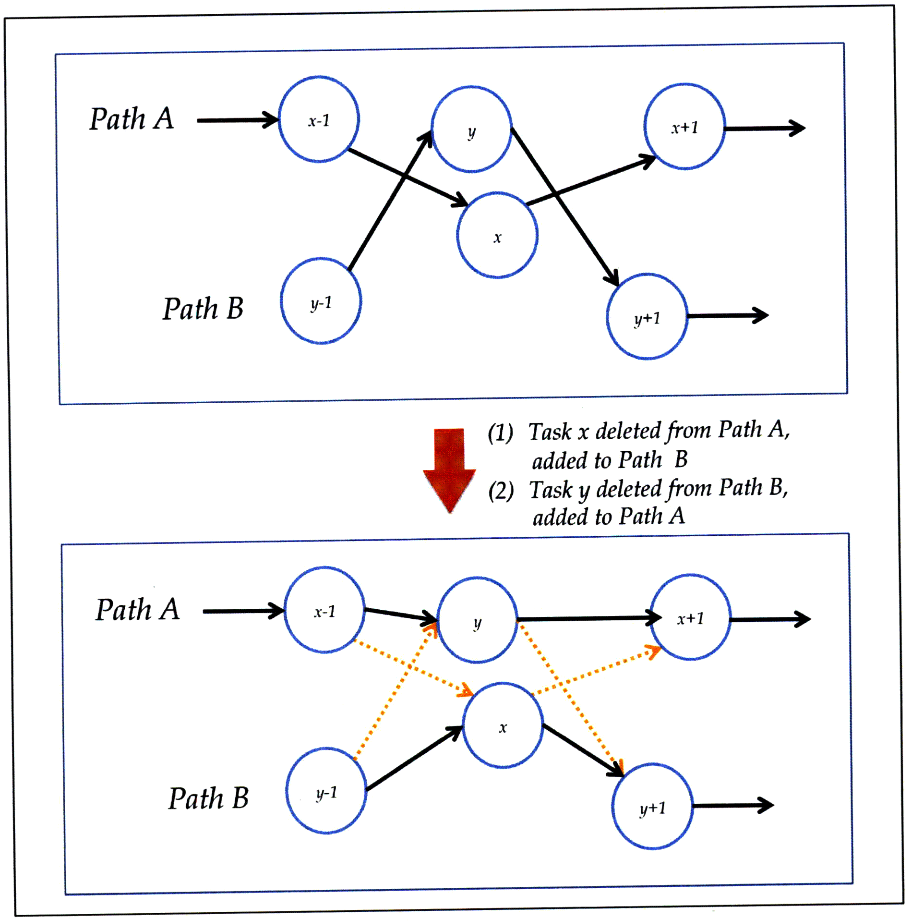
Example 4-4. *Consider the example in the text above. The first path, Path A, has twelve tasks. The fourth task in the path, Task y, is being considered for deletion and insertion into another path. Task y is at the 25th percentile of Path A. The path that Task y might be inserted into is Path B. Path B has eight tasks, therefore, the 25th percentile is the second task. Task y is inserted into the second spot in Path B. The feasibility of both Path A and B are checked; if feasible, the new durations are calculated. If the cumulative new duration is less than the previous duration, then Task y remains as the second task in Path B. Figure 4.5 illustrates this example.*

4.2.3.2.3 2-Exchange

The 2-Exchange method is also an inter-path method similar to the Deletion-Insertion method. In the 2-Exchange method, two tasks are switched between two paths. This is equivalent to deleting two arcs in from two separate paths and replacing them with two new arcs.

As with the 2-opt and Deletion-Insertion, the paths can become infeasible due to the time window constraints on the arcs. The method is therefore applied in same way as Deletion-Insertion. When considering which tasks to exchange between the routes, tasks in placements proportionally within the same percentile of the each other are considered. Example 4-5 illustrates this method.

Example 4-5. *The 2-Exchange method exchanges tasks between two paths. Consider exchanging Task x from Path A with Task y from Path B. Assume that Task x is the fourth task in Path A which has twelve tasks, and Task y is the second task in Path B which has eight tasks. Therefore, they are proportionally placed in the same percentile of each path. Task x is then deleted from Path A and Task y is inserted into the 4th placement in Path A. Task y is deleted from Path B and Task x is inserted into the 2nd placement in Path A. The feasibility of both paths is checked. If both are feasible, then the new durations are calculated. If the cumulative new duration is less than the previous duration, then the exchange is maintained. Figure 4.6 illustrates this example.*


Path A
x-1
y
x+1
x
Path B
y-1
y+1
(1) Task x deleted from Path A, added to Path B
(2) Task y deleted from Path B, added to Path A
Path A
x-1
y
x+1
x
Path B
y-1
y+1

4.6 2-Exchange Example

4.2.3.3 Insertion Phase

The Insertion Phase applies an insertion heuristic method to the path. In this phase, it is attempted to find a feasible location to insert each task that is not previously included in the set of paths, P.

The phase begins by determining which tasks are not included in any of the paths; this is the set T'. The task locations for the tasks that were not included are put in the set L. The set is then ordered by value and represented by L'. Starting with the highest value task location, it is attempted to insert the task into each of the paths. When a feasible placement for the task is found, the task is inserted into the path. All locations from the path associated with the tasks are then deleted from L'.

The method continues until as many tasks have been inserted as are feasible. This phase is summarized in the following steps. The index m is used to denote the placement in a path; p is used as an index for the paths in set P:

(1) Collect set of tasks not included in any path in *P*; this is *T'*.

(2) Get the set of locations associated with the tasks in *T'*; this is *L*.

(3) Sort *L* by value to get *L'*.

(4) Loop through *L'*, starting with $l = 1$. Initialize $p = 1, m = 1$.

(i) Check if feasible to insert location *l* into path *p* in placement *m*.

(ii) If feasible, insert, delete *l* and all other locations associated with the same task from *L'*, and return to (4), $l = l + 1$. If infeasible, return to (i), $m = m + 1$.

4.2.3.4 Location Swap Phase

The Location Swap Phase is not part of the 3PAA proposed by Miller, but was added to the algorithm to ensure that the tasks are completed in the most valuable location possible. In this phase, locations can be swapped out for higher value locations of the same task. For example, if the first task in Path A is currently performed at a location with a value of 10 but it can be performed at a location with a value of 15, then the higher value location is swapped for the lower value location, if feasible. The swap is considered for each location in each path.

The notation used is the same as in the previous subset. In addition, the set *H* represents the set of task locations with higher value than the current location for the task that is included in the path. The index *h* is used as an index for the locations in set *H*. Recall that *P* is the entire set of path and *p* is used as an index for the paths in *P*. The steps of the method are as follows:

(1) Loop through the paths in *P*. Start with $p = 1$.

(2) Loop through all tasks in *p*. The placement is denoted by $m = 1$.

(3) Find all locations of higher value for the task in placement *m* of path *p*; this is *H*.

(4) Loop through *H*; $h = 1$.

(i) Attempt to swap location *h* with location *m* in *p*.

(ii) If feasible, swap. If infeasible, $h = h + 1$.

4.2.3.5 Review of Composite Generation

The Composite Generation Step of COPA, described in the preceding sections, is the most complex part of the algorithm. Multiple heuristics are applied to each path in an organized manner to produces an improved solution. An overview of composite generation is summarized in the following steps:

(1) Construction Phase: Construct initial paths.

(2) Improvement Phase:

(i) Iterate 2-Opt for a set number of exchanges or until no improvement.

(ii) Iterate Insertion-Deletion for a set number of exchanges or until no improvement.

(iii) Iterate 2-Exchange for a set number of exchanges or until no improvement.

(5) Insertion Phase: Insert as many unperformed tasks as possible.

(6) Location Swap Phase: Swap out task locations for locations with higher value.

The result of the Composite Generation Step is a set of composites that represent paths and are used in the third step of the algorithm. The paths were created to be performed by a specific type of UAV; and each path has an complete set of ordered tasks that define the location that a task should be performed at, the start times, and end times for each observation. The next section describes how these composites are used in the FPCVF described in Section 4.1.3 to create as operations plan for the system of UAVs.

4.2.4 *Step III: Composite Variable Linear Program*

The last step of COPA is to solve FPCVF introduced in Section 4.1.3. The result of the linear program is a set of full path composites, one for each UAV in the system. The composites are incorporated into an operations plan for the system.

The purpose of the FPCVF is to find a *set cover* for the set of UAVs that is most valuable to the user. A set cover ensures that each UAV is assigned to a composite, so that a complete operations plan is created by the set of composites. Each time Step I and II are completed, the result is a set covering set of paths. However, the set of paths created during the first run may not be the most valuable; in fact, the FPCVF chooses the best combination of paths that is a set cover for the UAV regardless of the run from which the composite paths were created.

The results of the FPCVF depend on the number of composites created during Step I and II. With more composites, the FPCVF has more feasible sets of paths that could be assigned to the UAVs. The number of composites is determined by the number of times that Step I and II are run; this number is determined before running the algorithm. Example 4-6 provides an example of the linear programming step in the algorithm.

Example 4-6. *This example looks at a scenario with two types of UAVs: Type 1 and Type 2. There is a single UAV of each type. There are six tasks in the scenario; each task can be performed at two locations. The notation for the task locations is (i, j) where i denotes the task and j denotes the location. The following table summarizes the composites path and UAV types:*

Composite Variable	*UAV Type*	*Path*	*Value*
γ_1	1	(6,1) → (5,1)	80
γ_2	1	(3, 1) → (4,1)	40
γ_3	1	(2,1) → (1,2)→(4,2)	100
γ_4	2	(3, 1) → (4,1)	40
γ_5	2	(3, 1) → (5, 1)	60
γ_6	2	(6, 2) →(5,1)	70

Table 4.1 Example 4-6 Composites

Using these composites, the resulting FPCVF is:

$$\text{Maximize} \quad 80\gamma_1 + 40\gamma_2 + 100\gamma_3 + 40\gamma_4 + 60\gamma_5 + 70\gamma_6 \quad (4.9)$$

$$\text{Subject to} \quad \gamma_1 + \gamma_2 + \gamma_3 \le 1 \quad (4.10)$$

$$\gamma_4 + \gamma_5 + \gamma_6 \le 1 \quad (4.11)$$

$$\gamma_2 + \gamma_4 + \gamma_5 \le 1 \quad (4.12)$$

$$\gamma_2 + \gamma_3 + \gamma_4 \le 1 \quad (4.13)$$

$$\gamma_1 + \gamma_5 + \gamma_6 \le 1 \quad (4.14)$$

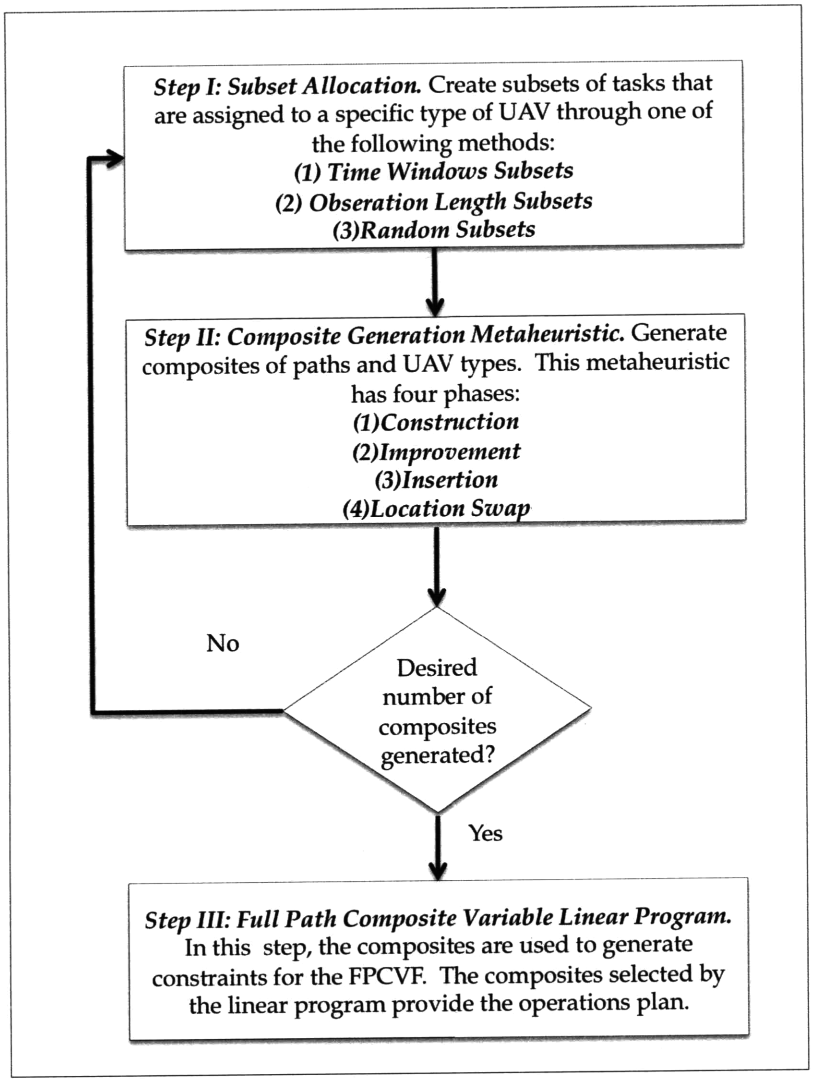
$$\gamma_1 + \gamma_6 \le 1 \quad (4.15)$$

$$\gamma_i \le 1 \quad \forall i = 1..6 \quad (4.16)$$

The objective value, 4.9, indicates that the objective is to maximize the total value of the composites chosen by the linear program. Constraints 4.10 and 4.11 ensure that each UAV is assigned a single path. Constraint 4.12 ensure that Task 3 is only included in the composite paths a single time; essentially, that the task is only performed once. Notice that Task 3 is included in the paths for Composites 2, 4, and 5. Constraints 4.13, 4.14, and 4.15 have the same function for Tasks 4, 5, and 6, respectively. Constraints for Task 1 and Task 2 are not needed, because they are included in only one composite each.

4.2.5 *Overview of Algorithm*

This section provides a summary of COPA as it has been described through the preceding sections. Because the algorithm includes loops, the algorithm is shown in a flow chart in Figure 4.7. In the figure, the steps are labeled as Subset Allocation, Composite Generation, and the Full Path Composite Variable Linear Program. The result of the algorithm is an operations plan for the UAVs.



Step I: Subset Allocation. Create subsets of tasks that are assigned to a specific type of UAV through one of the following methods:
(1) Time Windows Subsets
(2) Observation Length Subsets
(3)Random Subsets

Step II: Composite Generation Metaheuristic. Generate composites of paths and UAV types. This metaheuristic has four phases:
(1)Construction
(2)Improvement
(3)Insertion
(4)Location Swap

Desired number of composites generated?

Step III: Full Path Composite Variable Linear Program. In this step, the composites are used to generate constraints for the FPCVF. The composites selected by the linear program provide the operations plan.

Figure 4.7 Overview of Composite Operations Planning Algorithm

4.3 Implementation of COPA

To test the ability of COPA to produce optimized operations plans, the algorithm is implemented in NetBeans 6.1 using Java. To perform the last part of the algorithm, the FPCVF, the optimization software CPLEX 11.0 is called from Java. The implementation is useful for testing various cases with differing characteristics. An important aspect is the ability to test COPA on test cases with large numbers of UAVs and tasks.

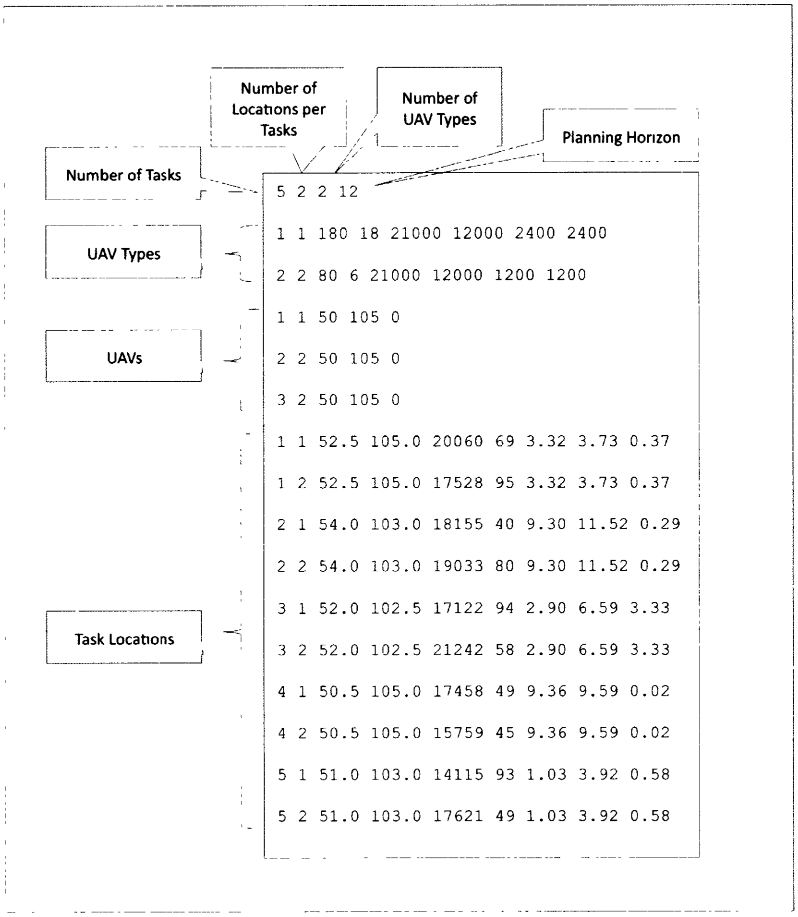
The implementation also allows for analysis on the runtime of the algorithm. The runtime analysis can also be compared to the runtime of the MIP introduced in the previous chapter to determine which method can solve large-scale problems in a reasonable amount of time.

This section introduces how data is inputted into the COPA software, how the test sets are created, and discusses the output of the COPA software.

4.3.1 *Sample Input Data*

To implement COPA, data describing the UAVs and the tasks are input using Java. The data is stored in a single text file that takes the form of the data in Figure 4.8.

The first line of the data file provides information about the test case: the number of tasks, the number of locations per task, the number of UAV types, and the planning horizon. The next block of data provides information about the UAVs types: speed, endurance, ceiling, floor, climb rate, and sink rate, and the individual UAVs' starting locations. Finally, the data describing the task locations follows: the task identification number, the location number, latitude, longitude, altitude, point value, early time window, late time window, and the necessary observation time. The example in Figure 4.8 illustrates the data.


Number of Locations per Tasks
Number of UAV Types
Planning Horizon
Number of Tasks
UAV Types
UAVs
Task Locations

```
5 2 2 12

1 1 180 18 21000 12000 2400 2400

2 2 80 6 21000 12000 1200 1200

1 1 50 105 0

2 2 50 105 0

3 2 50 105 0

1 1 52.5 105.0 20060 69 3.32 3.73 0.37

1 2 52.5 105.0 17528 95 3.32 3.73 0.37

2 1 54.0 103.0 18155 40 9.30 11.52 0.29

2 2 54.0 103.0 19033 80 9.30 11.52 0.29

3 1 52.0 102.5 17122 94 2.90 6.59 3.33

3 2 52.0 102.5 21242 58 2.90 6.59 3.33

4 1 50.5 105.0 17458 49 9.36 9.59 0.02

4 2 50.5 105.0 15759 45 9.36 9.59 0.02

5 1 51.0 103.0 14115 93 1.03 3.92 0.58

5 2 51.0 103.0 17621 49 1.03 3.92 0.58
```

Figure 4.8 Sample Data

4.3.2 Test Set Development

Each test set requires a large amount of data to describe the necessary input. To automate the process of creating test sets, a program using random numbers was developed in MATLAB.

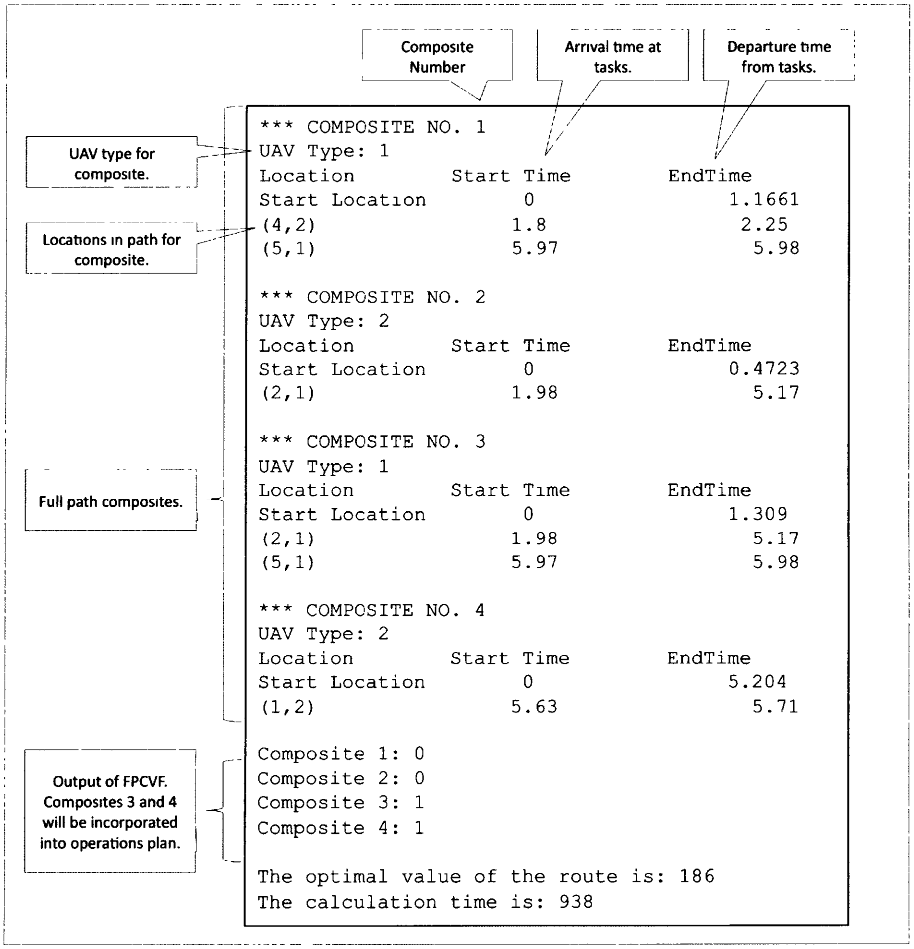
Some of the data was determined prior to the creation of the test set. The data describing the UAVs was determined based on information about operational UAV types that have been used for Earth monitoring missions. The starting locations for the individual UAVs are the predetermined locations for the command stations.

The data describing the tasks was autonomously created using a random number generator and mapping the number to a reasonable range. The distributions of the latitude, longitude, altitude, time windows, and observation times were uniformly distributed. The observation time was computed after the time window duration to ensure that duration of the time window was long enough to allow the UAV to complete the task.

To create different locations for each of the tasks that were in close proximity to each other, the values for the latitude, longitude, and altitude were perturbed slightly, using a separate random number generator. This ensured that the different locations associated with the same tasks were relatively close to each other. The time windows and observation time remained the same for each location.

4.3.3 Output

The output of the program provides the path and observation plan for the set of UAVs after solving the FPCVF. In addition, it outputs the information for all of the composites that are created, if the information is required. It also records the runtime of the algorithm, which times how long it took the computer to produces the path and observation plan, including data input and output. An example of the output of the COPA software in shown in Figure 4.9.



```
*** COMPOSITE NO. 1
UAV Type: 1
Location        Start Time        EndTime
Start Location       0               1.1661
(4,2)               1.8               2.25
(5,1)               5.97               5.98

*** COMPOSITE NO. 2
UAV Type: 2
Location        Start Time        EndTime
Start Location       0               0.4723
(2,1)               1.98               5.17

*** COMPOSITE NO. 3
UAV Type: 1
Location        Start Time        EndTime
Start Location       0               1.309
(2,1)               1.98              5.17
(5,1)               5.97              5.98

*** COMPOSITE NO. 4
UAV Type: 2
Location        Start Time        EndTime
Start Location       0               5.204
(1,2)               5.63              5.71

Composite 1: 0
Composite 2: 0
Composite 3: 1
Composite 4: 1

The optimal value of the route is: 186
The calculation time is: 938
```

Composite Number
Arrival time at tasks.
Departure time from tasks.
UAV type for composite.
Locations in path for composite.
Full path composites.
Output of FPCVF. Composites 3 and 4 will be incorporated into operations plan.

Figure 4.9 Example Output

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Testing and Analysis

This chapter evaluates two methods proposed to solve the UAV Planner Problem: the Mixed-Integer Program (MIP) formulation presented in Chapter 3 and the Composite Operations Planning Algorithm (COPA) presented in Chapter 4. The first part of the chapter focuses on a comparison of the two methods.

The chapter then focuses on the runtime of COPA. Because the goal of this research is to find an efficient method to solve the problem, Section 5.2 analyzes the sensitivity of COPA's runtime to additional task and task locations. The ability of COPA to handle large-scale cases of the UAV Planner Problem is tested to determine if COPA is able to handle large test sets.

Section 5.3 follows a scenario in which operator input is incorporated into COPA. The results of this scenario illustrate how the algorithm uses operator input in the development of an operations plan. Section 5.4 looks closely at the operations plans developed for this scenario.

Section 5.5 utilizes two scenarios to analyze the performance of two concepts of operations: a centralized system and a decentralized system. In centralized operations planning, COPA is used to generate an integrated operations plan at a centralized command station for a system of UAVs that are deployed out of separate command stations. The planning for a centralized system is called *decentralized operations planning*. For a decentralized system, the operations plans for the UAV are generated separately for each command station, without knowledge of the other station's operations. The planning for this system is called *decentralized operations planning*. The scenarios demonstrate the advantages and disadvantages of the two concepts of operations.

5.1 Comparison of COPA and MIP

This section focuses on the performance of COPA and the MIP. To determine which method is better suited to solve the UAV Planner Problem in an operational context, the methods are compared by analyzing two characteristics: the runtime of the method and the value of the resulting operations plan.

The software discussed in Section 4.3 is used in the analysis of COPA. The COPA software is implemented in Java and calls CPLEX 11.0 to solve the FPCVF. COPA was developed to have a short runtime. It uses a heuristic method that does not search the entire set of solutions, and, therefore, can identify a solution more quickly than an algorithm that considers every solution. However, because it does not search the entire set of solutions, it cannot guarantee that the resulting operations plan is optimal.

The MIP analysis utilizes ILOG's OPL Studio 5.5 and CPLEX 11.0 to solve the linear program. The software uses an exact solution method to determine the optimal solution. The *optimal solution* is the "best" solution to the UAV Planner Problem; it is the solution that has the highest value to the user. In the worst case, the MIP searches the entire solution space to find the optimal solution. The solution space is defined as the entire set of solutions to an instance of the UAV Planner Problem. Therefore, the runtime for the MIP can be lengthy.

This section provides analysis and testing data to confirm that COPA is a quick algorithm that might produce sub-optimal operations plans. It also illustrates that cases with fifteen or more tasks are intractable using the MIP formulation.

5.1.1 Testing Method

To analyze COPA and the MIP, twenty pseudo-random test sets were created using the method described in Section 4.3.2. There are four test sets each for the following number of tasks: 5, 10, 15, and 20. The tasks in each test set can be performed at two locations each. Therefore, for a test set with 5 tasks, there are 10 total locations, or, equivalently, 10 nodes in the network.

Each test set has two types of UAVs. Four UAVs were included in each test set, two of Type 1 and two of Type 2. The UAV types are described in the following table:

Type	*Speed (mph)*	*Endurance (hr)*	*Ceiling (ft)*	*Floor (ft)*	*Climb Rate (ft/min)*	*Sink Rate (ft/min)*
1	180	18	21000	12000	2400	2400
2	80	6	21000	12000	1200	1200

Table 5.1: UAV Types

For COPA, the number of composite variables generated by the algorithm affects the results of both the runtime and optimality analysis. Therefore, it was determined that each test set would be run with the following number of composites: 24, 48, and 72. For each result, the number of composites generated is noted.

When testing the MIP, the *linear programming (LP) relaxation*, is also tested using the same test sets. The LP relaxation relaxes the binary constraints on the decision variables. Specifically, in the LP relaxation of the MIP presented for the UAV Planner Problem, the variables $perform_{(i,k),u,t}$ and $travel_{(i,k),(j,l),u}$ are allowed to take any value on the interval [0, 1]. Therefore, the LP relaxation does not guarantee an executable operations plan. However, it does provide an upper bound for the MIP's performance that can be useful for cases in which the MIP cannot confirm a solution.

5.1.2 Runtime Comparison

The goal of the runtime comparison is to determine whether COPA or the MIP is able to create an operations plan that solves the UAV Planner Problem in the shorter amount of time. To test this capability, the twenty test sets discussed in the previous section were solved using each method: COPA, the MIP, and the LP relaxation. Each test set was solved three times using COPA. First, COPA generated 24 composites; second 48 composites were generated; and third, 72 composites were generated.

It is expected for COPA to be the faster method. This is because COPA does not search the entire solution space to find the optimal solution. In fact, the COPA's solution might not be optimal; however, it does select the best solution out of those considered by the algorithm. Conversely, the MIP is an exact solution method that, when completed, provides the optimal solution. Table 5.2 provides the runtimes of each of the testing runs.

Size of Problem (Tasks, Locations)	COPA Runtime (s) 24 Comp.	COPA Runtime (s) 48 Comp.	COPA Runtime (s) 72 Comp.	MIP Runtime (s)	LP Relaxation Runtime (s)
(5,2)	0.81	1.17	1.376	2.93	0.26
(5,2)	0.734	0.817	0.938	20.53	0.25
(5,2)	0.613	0.526	0.848	7.42	0.25
(5,2)	0.617	0.679	0.99	16.28	0.25
(10,2)	0.357	0.98	0.913	3625	1.5
(10,2)	0.348	0.832	0.939	2640.53	1.5
(10,2)	0.398	0.456	0.49	6081	1.32
(10,2)	0.323	0.563	0.866	2732.16	1.81
(15,2)	0.82	1.099	1.163	7881.61*	28.53
(15,2)	0.825	0.941	1.088	1178.4*	954.8
(15,2)	0.414	0.626	0.764	19866.41*	1600.0
(15,2)	0.439	0.772	1.23	220.65	1292.8
(20,2)	0.764	0.748	0.989	4084.25*	651577.3
(20,2)	0.457	0.736	1.537	2912.10*	53.28
(20,2)	0.543	1.095	1.211	40092.43*	2376784.4
(20,2)	0.48	0.757	1.087	21816.79*	35.53
* = Time until best solution was found (unable to prove).					

Table 5.2 Runtime Analysis

The data presented in Table 5.2 indicates that, for each test set, the COPA algorithm solves the problem in the shortest amount of time. In fact, as the size of the test sets increased, the MIP was unable to confirm that the solution provided by the program was optimal, even after allowing the algorithm to search the solution space for a long period of time.

Table 5.3 shows the averages of the runtimes for each size of the problem. It is important to note that, for the test cases with 20 tasks, the MIP did not solve any of these problems to completion, which explains the decrease in average runtime between the test cases with 15 task and those with 20 tasks for the MIP.

Size of Problem	*Average COPA Runtime (s) 24 Comp.*	*Average COPA Runtime (s) 48 Comp.*	*Average COPA Runtime (s) 72 Comp.*	*Average MIP Runtime (s)*
(5, 2)	0.694	0.798	1.038	11.79
(10, 2)	0.357	0.708	0.802	3769.67
(15, 2)	0.625	0.860	1.061	10756.87
(20, 2)	0.561	0.834	1.206	3498.18

Table 5.3 Average Runtimes

The average runtimes support the claim that COPA solves the UAV Planner Problem in a shorter amount of time. For example, in test sets with 15 tasks that could be performed at two locations each, the average runtime for COPA with 72 composites generated is 0.802 seconds, while the average runtime for the MIP is 1 hour, 2 minutes, and 49.67 seconds. The average runtimes for COPA are shown in Figure 5.1; the MIP runtimes are not included.

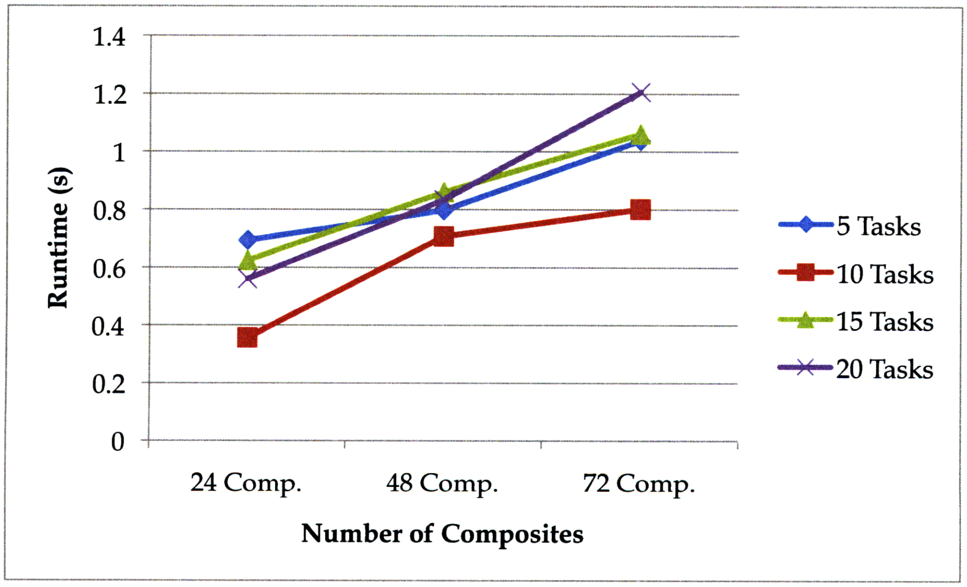


Figure 5.1 Average Runtimes for COPA

Another important aspect is the variance of the runtimes. In an operational situation, it would be detrimental to use an algorithm that might take significantly longer than expected; hence, the standard deviation of the runtime should be evaluated. Table 5.4 gives the standard deviations of the runtimes; this shows that the standard deviations of the runtimes for the MIP are orders of magnitude larger than the standard deviations of the runtimes for COPA.

Size of Problem	*Standard Deviation of COPA Runtime (s) 24 Comp.*	*Standard Deviation of COPA Runtime (s) 48 Comp.*	*Standard Deviation of COPA Runtime 72 Comp.*	*Standard Deviation of MIP Runtime*
(5, 2)	0.096	0.275	0.233	8.045
(10, 2)	0.031	0.241	0.210	1603.596
(15, 2)	0.229	0.205	0.206	9052.503
(20, 2)	0.140	0.174	0.239	17526.573

Table 5.4 Standard Deviation of Runtimes

5.1.3 Optimality Comparison

The objective for the optimality comparison is to determine how close the solution selected by COPA is to the optimal solution, which is provided by the MIP. By definition, the MIP, if it is able to find one, provides an optimal solution: the solution with the highest value to the user.

To determine the ability of COPA to generate solutions close to the optimal solution, COPA was applied to the twenty test sets discussed in Section 5.1.1 and used for the runtime analysis. Similar to the runtime analysis, COPA was applied to each test set three times: first, 24 composites were generated; second, 48 composites were generated; and, third, 72 composites were generated.

The value of the operations plan developed by COPA depends on the number of composites generated. With a greater number of composites, the FPCVF linear program can choose from a wider selection of composites. Therefore, it is possible that the solution selected by COPA has higher value if more composites are generated. This is equivalent to stating that, by generating more composites, a larger solution space is explored by the algorithm in selecting the best solution. Table 5.5 presents the data for the optimality comparison.

Size of Problem	*Value of COPA Solution 24 Comp.*	*Value of COPA Solution 48 Comp.*	*Value of COPA Solution 72 Comp.*	*Value of MIP Solution*	*Lowest Optimality Gap*	*Value of LP Relaxation Solution*	*Integrality Gap*
(5,2)	245	255	255	295	13.56%	295	0%
(5,2)	161	161	161	161	0%	201	19.90%
(5,2)	274	275	275	281	2.14%	281	0%
(5,2)	279	279	279	279	0%	344	18.90%
(10,2)	543	543	543	555	1.30%	670.25	17.20%
(10,2)	660	660	664	739	10.15%	756	2.25%
(10,2)	634	696	696	696	0%	764	8.90%
(10,2)	432	434	434	466	6.87%	558	16.49%
(15,2)	851	888	888	959*	2.10%	1007	4.77%
(15,2)	884	884	884	884*	0%	960	7.92%
(15,2)	920	960	960	992*	3.23%	1054	5.88%
(15,2)	959	959	959	974	1.54%	1031	5.53%
(20,2)	1207	1247	1265	703*	**	1382	49.13%
(20,2)	1102	1102	1102	784*	**	1280	38.75%
(20,2)	1138	1171	1190	1127*	**	1330	15.26%
(20,2)	1122	1125	1139	766*	**	1240	38.23%

* = Best solution; computer unable to prove.
** = Algorithm solution greater than MIP solution

Table 5.5 Optimality Comparison

The data in Table 5.5 indicates that COPA provided solutions with values that were close to the value of the optimal solution; the values are shown visually in Figure 5.2. Out of the nine cases where the MIP was able to confirm the optimality of the solution, COPA provided answers of equivalent value. The average optimality gap of the nine cases with confirmed optimal solutions; the average optimality gap was 3.95%. The *optimality gap* is the percentage that a solution deviates from the optimal solution.

Another significant observation is that, in the test sets with 20 tasks, the computer was unable to provide a solution to the MIP that was better than the COPA solution before it ran out of memory. The actual solution to the MIP would have been better than or equal to the COPA solution, by definition; however, because the computer was unable to find the solution, a better solution cannot be found using these methods.

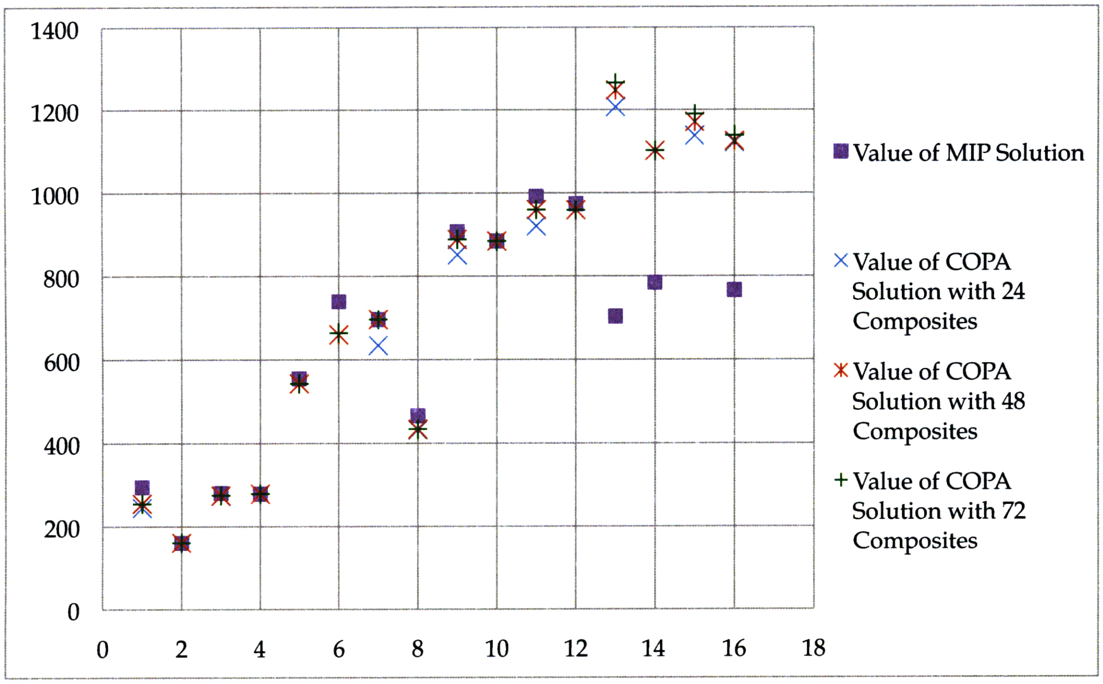


Figure 5.2 Comparison of Operations Plan Values for MIP and COPA

Appendix B provides a table with both the runtime and optimality results. The data in this table provides a succinct view of the runtime and optimality trade-offs of COPA and the MIP. The data shows that for the test sets with five tasks, the MIP did not have significantly longer runtimes, and was able to provide better solutions. However, as the test set got larger, the runtime of the MIP increased significantly; eventually, the test sets were intractable for the MIP.

Operationally, the UAV Planner Problem might include scenarios with large numbers of tasks. This suggests that COPA is better for the operational scenarios of the UAV Planner Problem. The runtime of algorithm did not increase significantly with the larger number of tasks and, for all test sets, provided a feasible, high value solution. The next section analyzes the runtime of COPA in greater detail, focusing on the impact that increasing the size of the test set has on the runtime of the algorithm.

5.2 COPA Runtime Sensitivity Analysis

The purpose of this section is to analyze the impact that the problem size has on the runtime of COPA. Runtime is important in the UAV Planner Problem and for dynamic re-planning of the problem. Additionally, the operational problem most likely includes large numbers of tasks, which increases the complexity of the problem and, hence, the runtime of solution methods. In this section, we discuss the impact that additional tasks and additional locations per task have on the runtime of COPA, and examine the ability of COPA to solve large-scale problems.

The analysis in this section is performed using test sets randomly generated, as discussed in Section 4.3.2. There is no continuity of test sets between the parts of the analysis; new test sets were generated for each section.

5.2.1 Impact of Additional Tasks and Locations

To determine the impact that additional tasks has on the runtime of COPA, pseudo-random test sets were used. The test sets contained numbers of tasks ranging from five to 50 tasks, in intervals of five. As expected the runtime of COPA increased with an increase in the number of tasks in the test set. The average increase in runtime was 0.02 seconds for an additional five tasks in the range of test sets analyzed.

Figure 5.3 illustrates the runtimes of the test sets. The figure shows the increase in runtime as the number of tasks increases. Also, the figure shows that the variance of the runtime increases with an increase in the number of tasks.

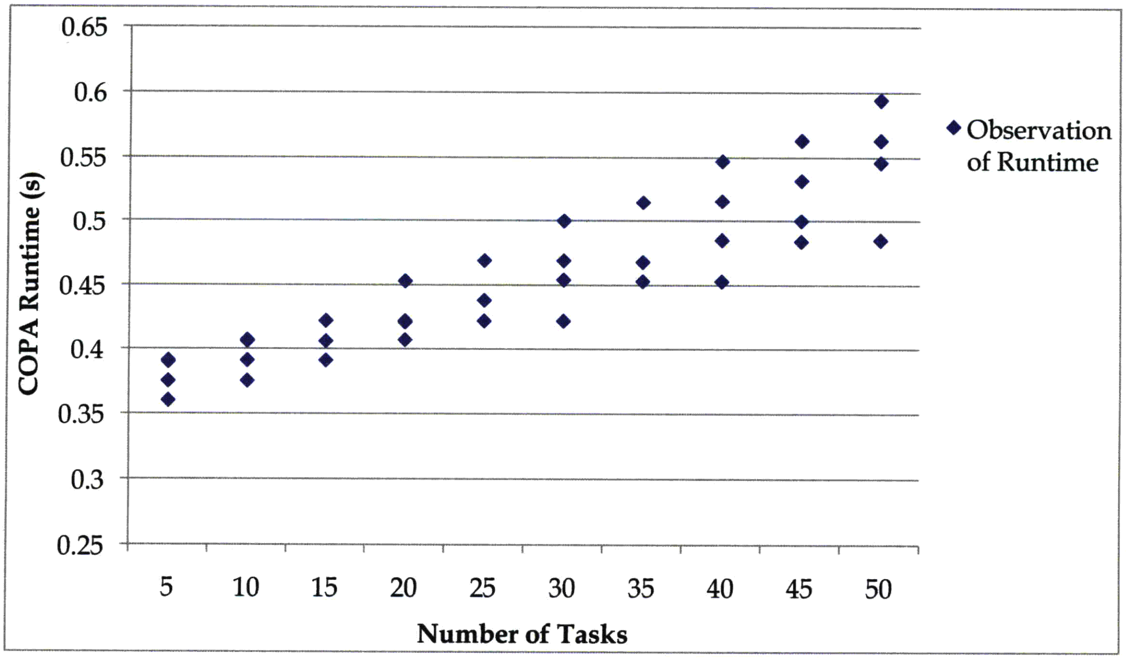


Figure 5.3 Impact of Additional Tasks on COPA Runtime

The impact of additional locations per task was tested using pseudo-random test sets with 10 tasks. The number of locations per task in the test sets ranged from two to 10 in intervals of 2. Therefore, the first test set had 10 tasks that could be performed at two locations each, the second test set had 10 tasks that could be performed at four locations each, and so on.

The testing concluded that additional locations per task affected the runtime by adding an average of 0.1 seconds per additional two locations in the range of test sets analyzed. The results are illustrated in Figure 5.4.

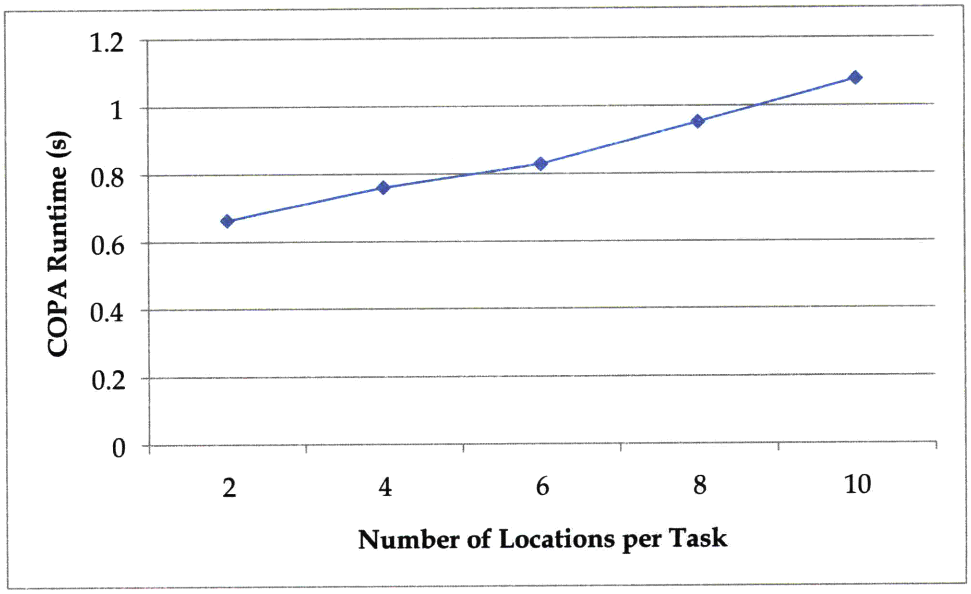


Figure 5.4 Impact of Additional Locations per Task on COPA Runtime

5.2.2 Large Size Test Cases

Operationally, the UAV Planner Problem can become very large. To understand how COPA handles large-sized problems, the algorithm was tested using test sets with an increasing number of tasks. Each test set had 15 UAVs, five of Type 1 and 10 of Type 2 as described in Table 5.1. The planning horizon for the test sets was six hours.

The testing began with 100 tasks that could be performed at 10 locations each; therefore, the number of nodes in the test set was 1,000. Then, 100 tasks were added for each test set. The largest test set added had 500 tasks that could be performed at 10 locations each for a total of 5,000 nodes in the problem. Tests with more tasks were not performed before the publishing of this thesis.

During the testing, COPA generated 30 composites for each test set. This number was determined before testing. Generating additional composites would increase the length of the runtime; therefore, there is a balance between generating composites and maintaining a reasonable runtime.

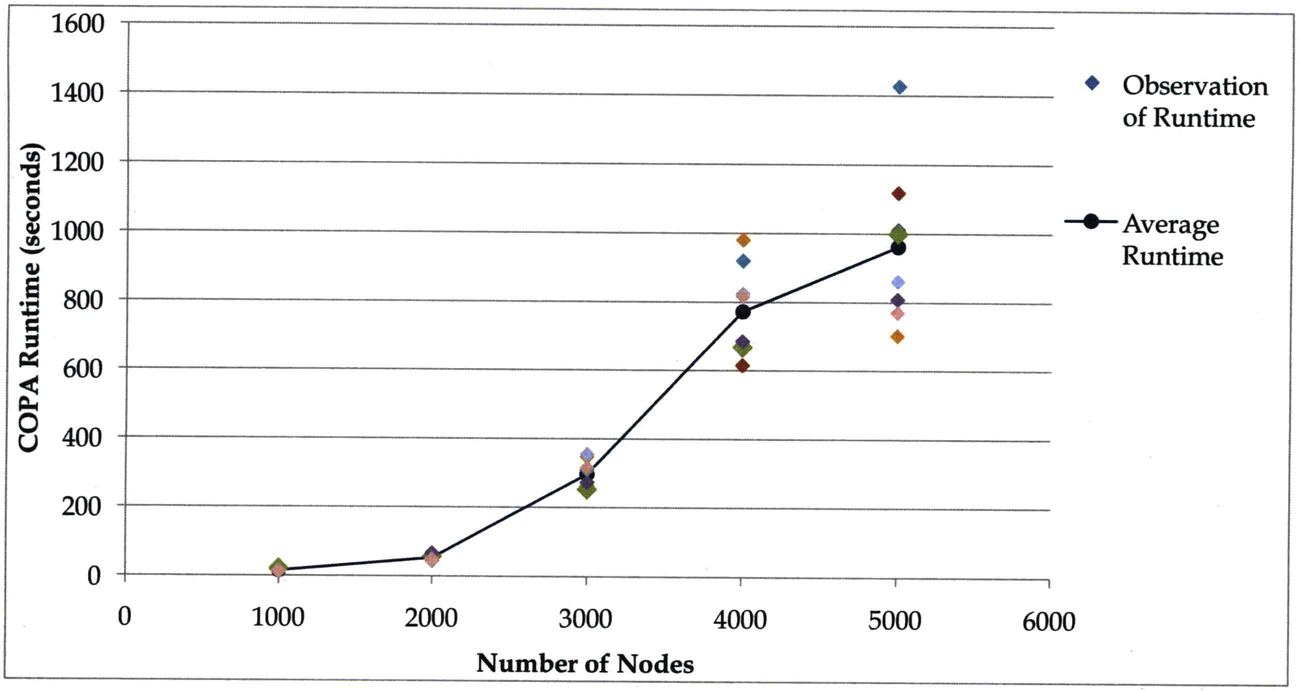


Figure 5.5 Runtimes of Large Test Sets

Figure 5.5 shows the runtimes of the large test sets; the runtimes are displayed for eight test sets of each size. The trend line connects the average runtimes.

COPA was able to handle these large test sets in a reasonable amount of time. The longest runtime was 23 minutes and 46.4 seconds for a test case with 500 tasks that could be performed at 10 locations each. This runtime is reasonable for an operational situation where the planning horizon is six hours.

5.3 Operator Input Scenario

Part of the UAV Planner Problem is to allow the operator to provide input to incorporate into the operations plan. COPA includes operator input by the operator assigning tasks to be performed by a specific UAV type before initiating the algorithm; this is called *pre-assigning* a task.

The input is used in the Subset Allocation step of COPA. The pre-assigned tasks are put into the subsets desired by the operator. The algorithm continues without alteration until the FPCVF is solved. After determining which composites produce the best operations plan, the operator is provided with multiple plans to choose from, with the most valuable set of composites highlighted.

The operator involvement in COPA is illustrated through a scenario. This scenario has six tasks that can be performed at two locations each. The locations differ only in altitude. There are two UAVs in this example; there is one UAV of Type 1 and one UAV of Type 2 as described in Table 5.1. Therefore the operator can pre-assign tasks to be completed by either type of UAV. The UAVs are centrally controlled and are dispatched from the same initial location. Figure 5.6 illustrates the scenario in two dimensions.

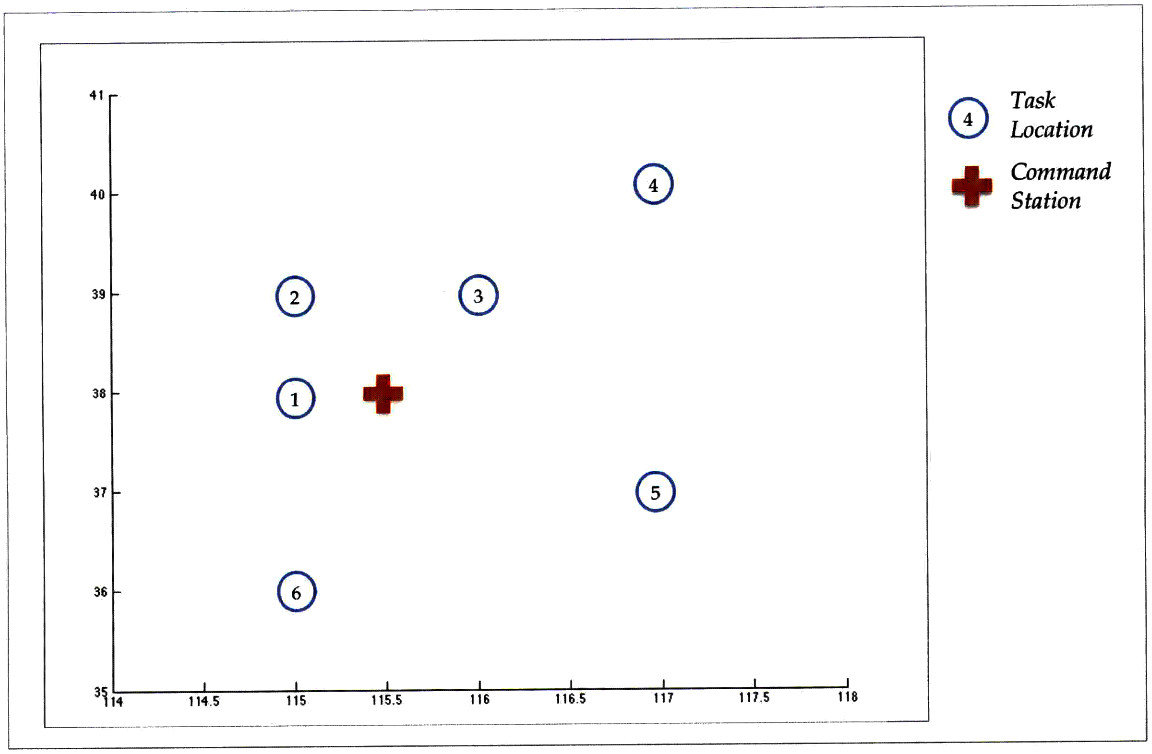


Figure 5.6 Tasks for Operator Input Scenario

For this example, COPA develops operations plans with operator input and without operator input and compares the operations plans developed by COPA in both cases.

The operator input is the following:

Set of tasks to be performed by UAV of Type 1: {1, 6}
Set of tasks to be performed by UAV of Type 2: {2, 4}

The pre-assigned tasks were chosen arbitrarily for the purpose of this illustration. These pre-assignments were put into COPA and the resulting operations plan includes the pre-assigned UAV types performing the tasks. This operations plan, which has a value of 350, is shown in Figure 5.7.

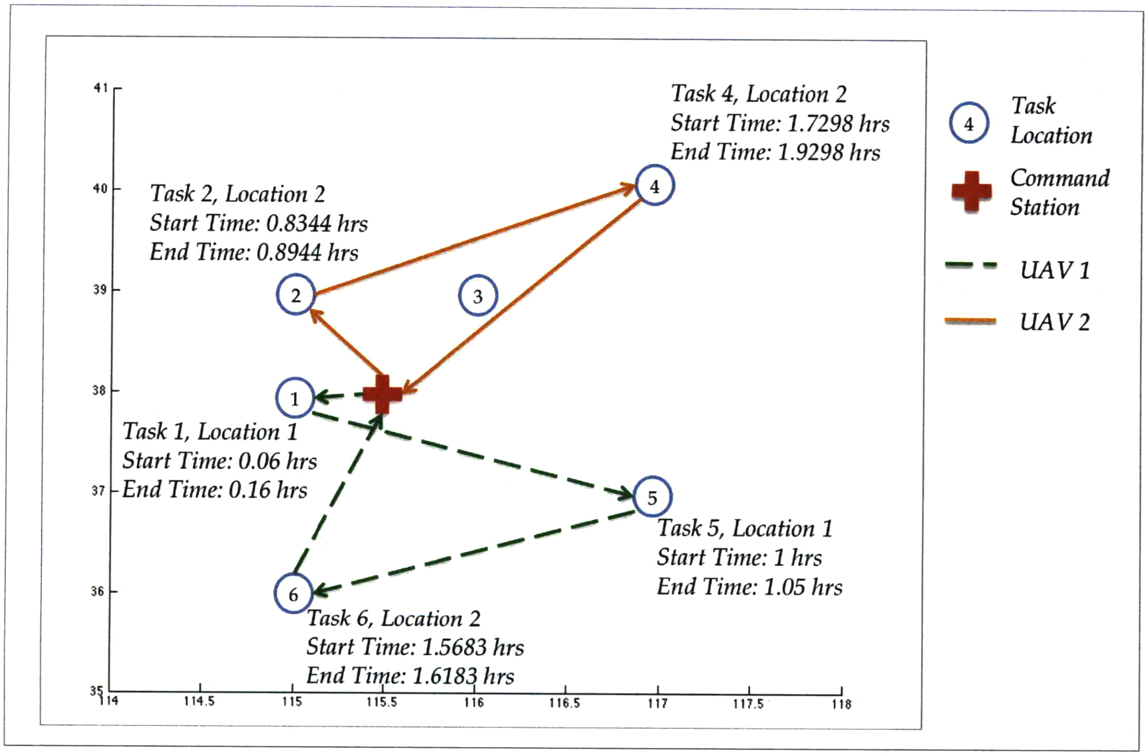


Figure 5.7 Operations Plan containing Pre-Assigned Tasks

An operations plan for the same scenario was also developed using COPA without any tasks pre-assigned. The resulting operations plan is shown in Figure 5.8. COPA was able to develop an operations plan with a higher total value of 360.

In this example, COPA was able to find a more valuable operations plan than with the operator input. Both options would be available to the operator before he or she decides which operations plan to execute. The section operations plan, with no pre-assigned tasks, would be highlighted because it has a greater value.

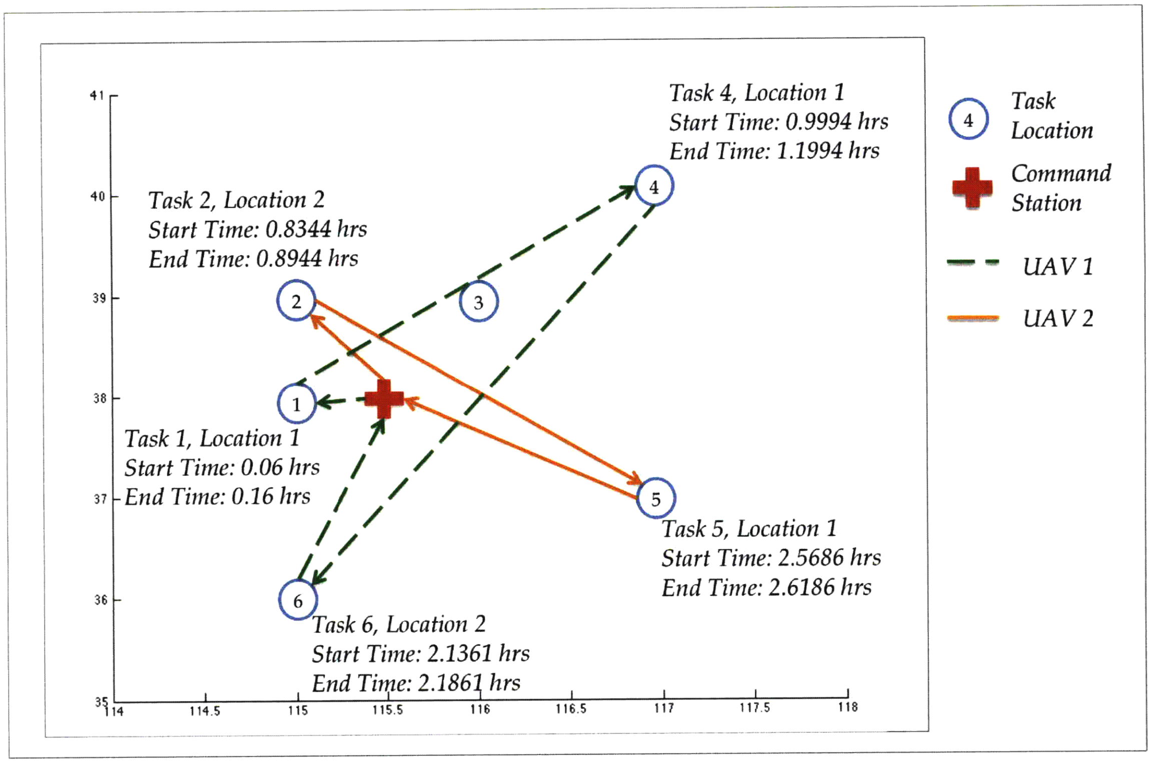


Figure 5.8 Operations Plan for Operator Input Scenario with No Pre-Assigned Tasks

5.4 Discussion of Operations Plans

This section will look closer at the operations plans developed by COPA in Section 5.4. The purpose is to improve our understanding of the plans developed by the algorithm. The reason for examining this particular example is because it includes a small number of tasks and UAVs, and, therefore, provides a good illustration.

To review, the scenario has six tasks that can be performed at two locations each. Two UAVs are in the scenario; one of Type 1 and one of Type 2, as described in Table 5.1. The planning horizon is three hours. The task data from the input file is shown in Table 5.6.

This analysis focuses on the operations plan developed by COPA without operator input, shown in Figure 5.8. The following paragraphs highlight some of the aspects of the scenarios that are not captured in the figures displaying the operations plans. Specifically, the impact of the following three aspects are discussed: value of the tasks, altitude, and time windows.

Task Location	*Latitude*	*Longitude*	*Altitude*	*Value*	*Early Time Window*	*Late Time Window*	*Observation Length*
(1, 1)	38	115	14000	60	0	1	0.1
(1, 2)	38	115	16000	60	0	1	0.1
(2, 1)	39	115	20000	100	0	1	0.06
(2, 2)	39	115	18000	80	0	1	0.06
(3, 1)	39	116	14000	60	0	1.5	0.2
(3, 2)	39	116	15000	20	0	1.5	0.2
(4, 1)	40	117	16000	80	0.4	1.5	0.2
(4, 2)	40	117	18000	40	0.4	1.5	0.2
(5, 1)	37	117	13000	100	1	3	0.05
(5, 2)	37	117	18000	70	1	3	0.05
(6, 1)	36	115	19000	80	1	3	0.05
(6, 2)	36	115	14000	70	1	3	0.05

Table 5.6 Task Data for Operator Input Scenario

First, the value of the tasks impacts the development of the operations plans. A concern about the operations plan shown in Figure 5.8 might be that the operations plan does not include Task 3, even though the path for UAV 1 seems to pass almost directly over the location of the task. The reason for this is the value of the tasks. If the UAV performs Task 3, which has a maximum value of 60, then it would be unable to perform either Task 4, with a value of 80, or Task 6, which is performed at location (6, 2) with a value of 70.

Second, it is important to understand the impact of altitude on operations planning, especially because it cannot be captured in the two-dimensional figures. For example, a concern about the operations plan shown in Figure 5.8 might be the crossing of the paths for UAV 1 and UAV 2. However, it is important to remember that altitude is also a factor in creating the operations plan. The altitudes of the task locations included in the plan for UAV 1 are:

(1, 1)	*Altitude: 14000 ft*
(4, 1)	*Altitude 16000 ft*
(6, 2)	*Altitude 14000 ft*

Contrast those altitudes with the altitude of the tasks in the plan for UAV 2:

(2, 2)	*Altitude: 18000 ft*
(5, 2)	*Altitude: 18000 ft*

Therefore, the plans do not actually cross each other, because the UAVs are operating at different altitudes.

The last concern that will be mentioned is a UAV crossing its own path. This is illustrated in the operations plans developed by COPA for the operator input scenario when the pre-assigning of tasks was included in the operations planning, which produced the operations plan shown in Figure 5.7. The reason for this is the time window constraints on the tasks. Time windows can constrain a task to be performed before another task, regardless of the efficiency of performing the tasks in that order. For example, if Task 1 and Task 5 from Table 5.6 are in the same path, then Task 1 must be performed earlier in the path than Task 5. This is because the time window for Task 1 ends one hour after the start of the operations, while the time window for Task 5 starts one hour after the start of operations. Therefore, the time windows constraints do not allow Task 5 to be performed before Task 1. Section 5.5.2 and Example 5-1 further discuss the impact of time windows.

5.5 Operational Forest Fire Scenarios

The purpose of this section is to provide the reader with an idea of how COPA functions in an operational scenario. In the scenarios, firefighters work in smaller teams, called *incident units*, placed in strategic locations around the perimeter of a wildfire. The leaders of these smaller groups are called *incident commanders*.

Two types of planning are discussed: centralized operations planning and decentralized operations planning. For centralized operations planning, a single command center has control over all of the UAVs in the system. The operations plan for each UAV is developed at the command center. While the UAVs are deployed from the location of the incident units, the plans are coordinated and all tasks are known before the operations plans are created.

For decentralized operations planning, incident commanders have control over a subset of the UAVs. The commanders oversee the development of the operations plans; therefore, the plans are not coordinated. The operations plans are developed knowing only the tasks generated by the incident group. The incident commander, and hence the operations planner, for the incident group are not aware of the tasks generated by other incident units.

In this section, two scenarios are presented using each operational concept. In the first scenario, the incident units and the tasks generated by the units are geographically separated. In the second scenario, the incident groups are stationed close to each other and their tasks are intermingled. Operations plans for both scenarios are developed by COPA using both centralized and a decentralized operations planning. This section illustrates the operations plans developed by COPA in each of these situations. Additional information on the operations plans, including the times that the UAVs should be at each task, is in Appendices C through G.

5.5.1 Improvement of Subset Allocation Methods

During the operational scenario analysis, it became apparent that there was a need to improve the methods for the Subset Allocation step of COPA. In previous analysis, the UAVs were deployed from identical starting locations. For this analysis, the methods described in Section 4.2.2 produced subsets resulted in high value full path composites.

In the operational scenarios, COPA was able to develop operations plans using these methods; however, it was apparent that the operations plans could be more valuable to the user. The subset allocation was not taking into account the initial location of the UAVs. Because of this, the subsets might contain tasks that are spatially distributed, making it difficult for a UAV to perform many of the tasks. Therefore, new methods were developed to be able to build higher valuable operations plans.

The new methods are based on the existing methods described in Section 4.2.2: observation length subsets, time window subsets, and random subsets. The methods were altered by adding an initializing step that takes into account the initial location of the UAV types. The step first assigned tasks to subsets depending on the initial locations of the UAVs type. The new method follows the following steps:

(1) Find the set of all initial locations for the UAVs; this is set I.

(2) Partition tasks into subsets based on distance from the initial locations in I. The set C_i is the set of tasks that is closest to an initial location i.

(3) Build subsets according to existing methods, within the subset of tasks C_i.

(4) Combine subsets that are for UAVs of the same type.

The new methods were incorporated into COPA for the following analysis. The value of the resulting operations plans increased with the new methods in place.

5.5.2 Scenario I: Geographically Separated Units

In this scenario, two incident units are geographically separated. The units, Unit A and Unit B, are working to contain the same fire, but are located on different sides of the perimeter of the fire. This scenario is analyzed using both centralized and decentralized operations planning.

The scenario begins with the generation of tasks by both Unit A and Unit B. These tasks create separate test cases for the UAV planner problem. The tasks generated by the units are shown in Figure 5.9. The figure illustrates that the task locations are geographically separated. For this scenario, the locations at which the tasks can be performed at differ only in altitude. Therefore, the illustration below is unable to show the separate locations, because it is two-dimensional.

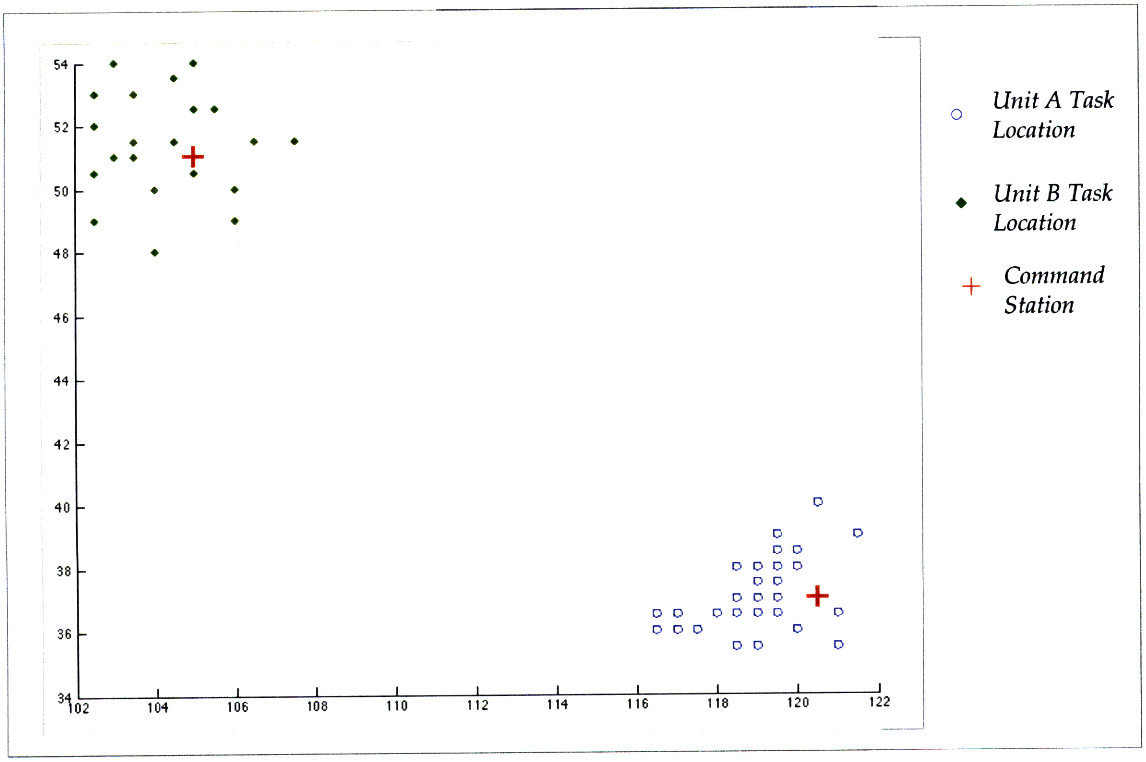


Figure 5.9 Scenario I Tasks

Coordination between the incident units might not result in a more valuable operations plan, because the tasks are geographically separated and it is impractical for the UAVs to travel between the sets of tasks. In fact, the best plans might be identical, because it is impractical for the UAVs to travel between the tasks generated by the two units.

Each unit has a single UAV of Type 1 and two UAVs of Type 2, as described in Table 5.1. For the decentralized operations planning, there were 36 composites generated for each unit, for a total of 72. In the centralized operations planning, 72 composites were generated.

5.5.2.1 Decentralized Operations Planning to Scenario I

First, operations plans are developed using decentralized operations planning. COPA is run individually for both Unit A's and Unit B's set of tasks. The command station is the location from which the UAVs are controlled; in a decentralized scenario, each incident unit has its own command station. This is not necessarily the location from which the UAVs are dispatched, but the term is used to denote the place that controls the UAVs.

The COPA software is first used to create an operations plan for the UAVs under the control of Unit A. Then, the software is used to create an operations plan for the UAVs under the control of Unit B. The operations plans developed by COPA for Scenario I are shown in Figure 5.10.

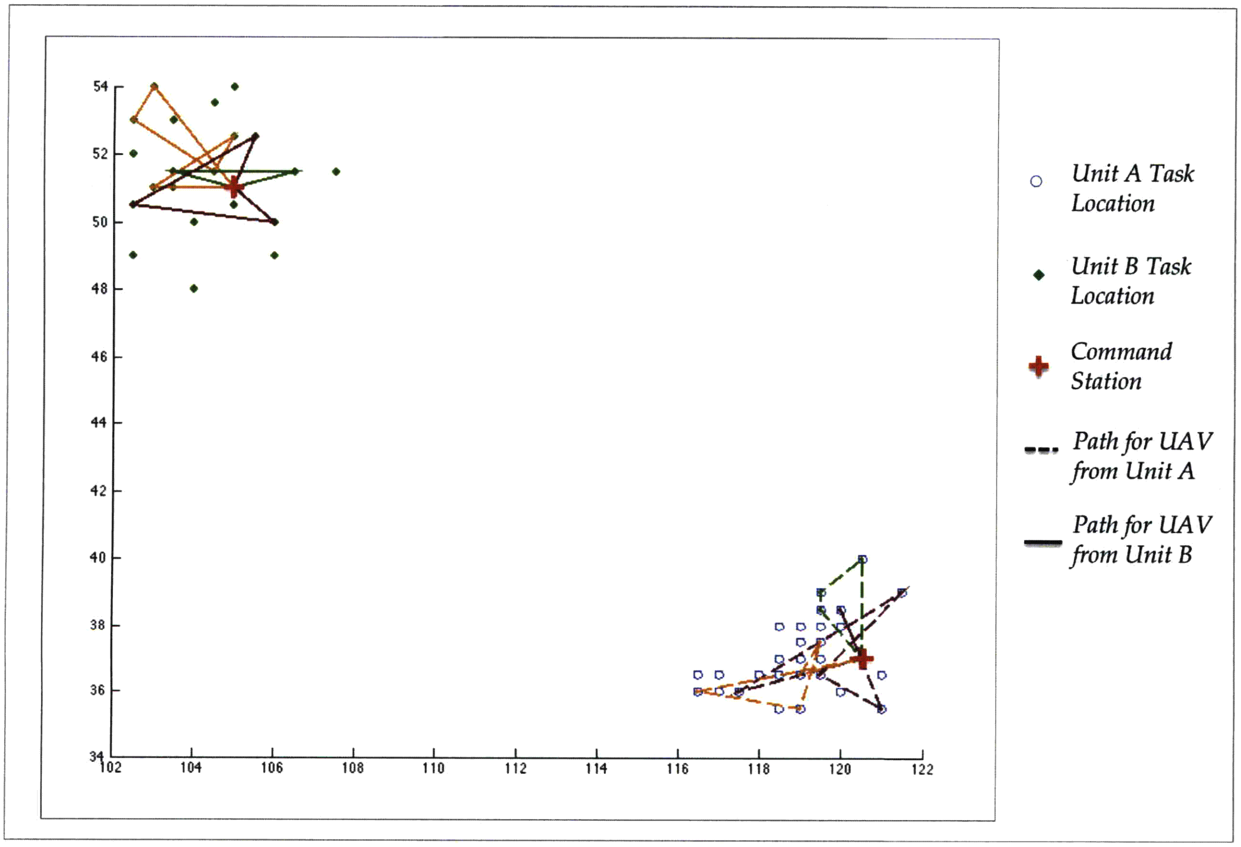


Figure 5.10 Decentralized Operations Plan for Scenario I

The value of the operations plan for Unit A was 986; the value of the operations plan for Unit B was 766. Therefore, the total value of the operations plans for Scenario I using decentralized operations planning was 1752.

5.5.2.2 Centralized Operations Planning for Scenario I

For centralized operations planning, the operations plans are developed at a central command center; however, the UAVs are still dispatched from the location of the incident units.

To use COPA for centralized planning for Scenario I, the task data for both units was input to the COPA software. The combined test case had 50 tasks and six UAVs. The results from the centralized operations planning is shown in Figure 5.11. The operations plan had a total value of 1770.

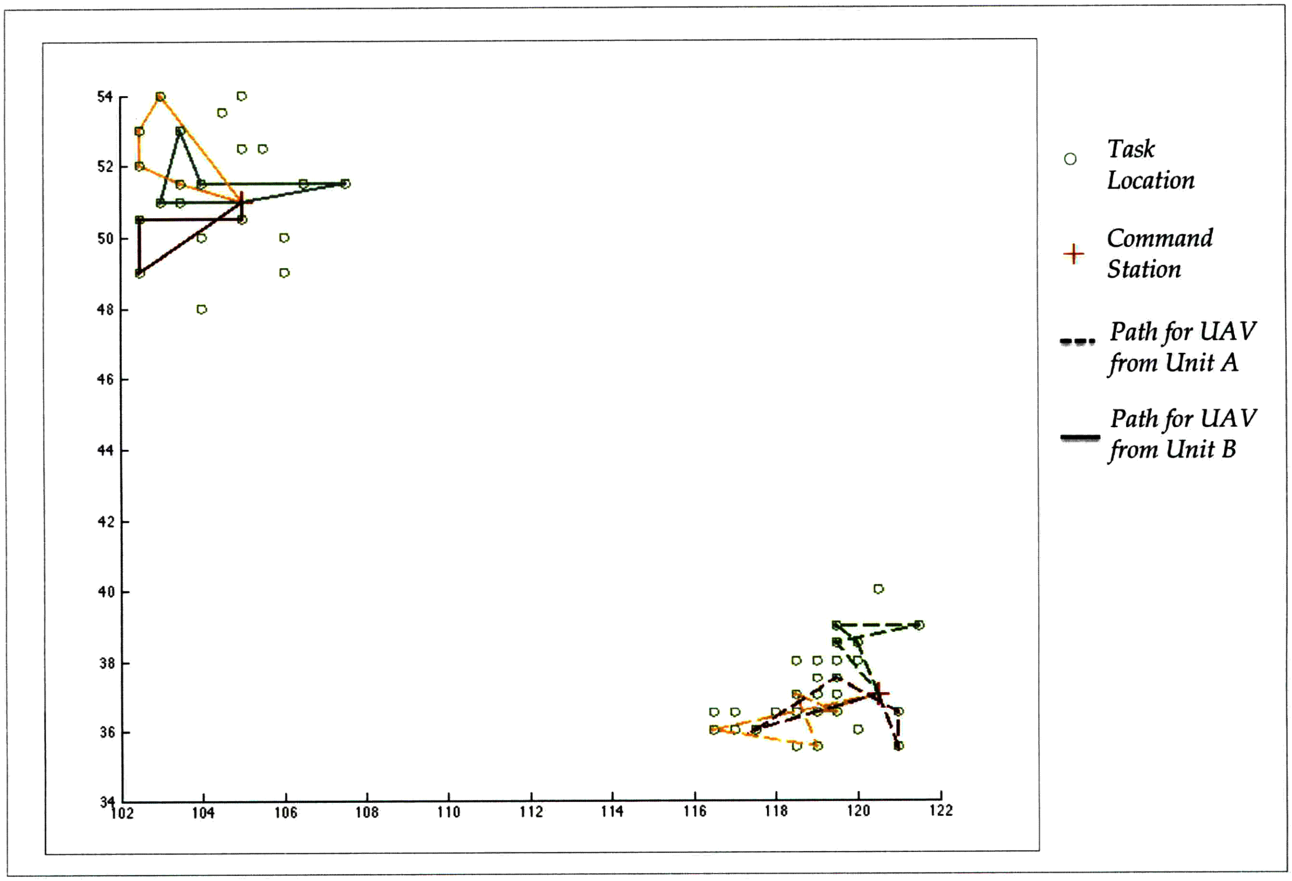


Figure 5.11 Centralized Operations Plan for Scenario I

5.5.2.3 Comparison between Decentralized and Centralized Planning for Scenario I

As Figures 5.13 and 5.14 illustrate, the UAVs for Unit A and Unit B only performed the tasks generated by their respective units for both decentralized and centralized operations planning. While this was expected due to the distance between the tasks generated by the units, it was not expected for the operations plans to differ. The centralized operations plan had a higher value than the cumulative value of the decentralized operations plan, despite the fact that it would be possible to produce the centralized operations plan through decentralized planning.

The difference in the operations plans is a result of the Subset Allocation step of COPA. If the same subsets were presented to COPA in both centralized and decentralized planning, then the result would be identical. However, the methods used for subset allocation produced differing subsets; specifically, for the subsets constructed using a random number generator.

The operations plans from both types of planning are similar. Both operations plans include 23 of the 50 tasks in the test set. In both plans, 12 of the tasks are from Unit A with the remaining 11 from Unit B. The values are fairly close: the cumulative value of the decentralized operations plans is 1752 and the value of the centralized operations plan was 1770.

The paths in the operations plans may not seem efficient, because the paths cross each other. As discussed in Section 5.4, this is an impact that time windows can have on operations planning. While it may be more efficient to perform tasks in a different order that may cause the performance of the tasks to no longer be valuable to the user. Example 5-1 illustrates how this happens in the case of Unit B's operations plan.

Example 5-1. *This example illustrates why the operations plans created by COPA include paths that cross each other. Specifically looking at the operations plan for Unit B in Scenario I using centralized operations planning, one of the paths crosses itself. The path is shown in Figure 5.12.*

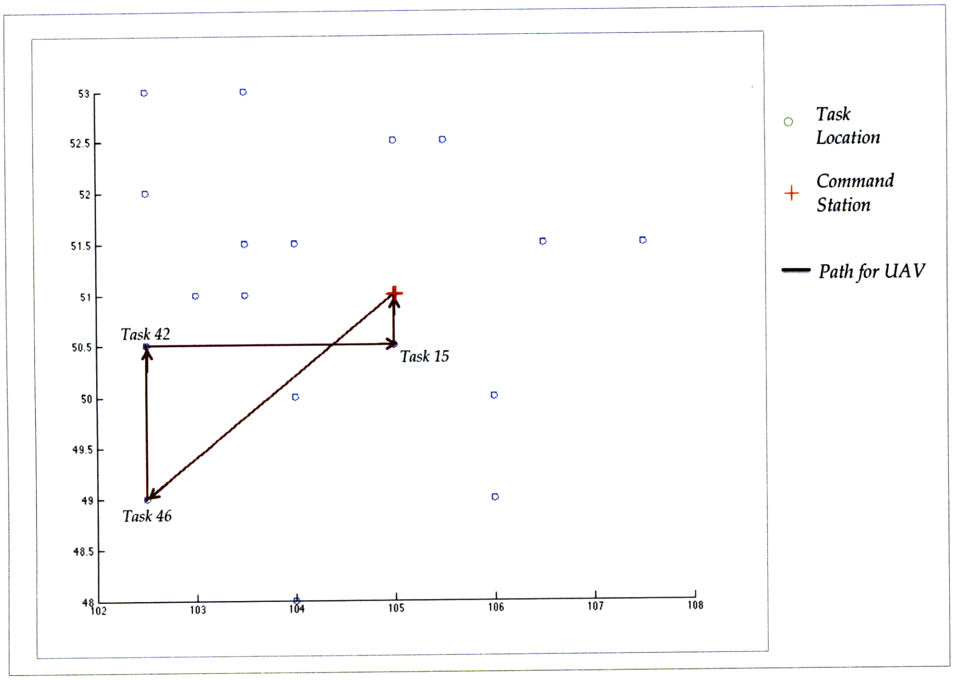


Figure 5.12 Example Path

The tasks completed by the UAV of Type 2 are, in order, Task 46, Task 42, and Task 15. The time windows for these tasks force the tasks to be completed in this order. The following table provides the time windows. The time windows are given in hours after the start time of the operations plan:

Task Number	***Early Time Window***	***Late Time Window***
46	*1.08*	*3.84*
42	*6.24*	*8.25*
15	*9.36*	*9.59*

Table 5.7 Example 5-1 Time Windows

A path that contains the tasks in the order {15, 46, 42} is more efficient than the current order, which is {46, 42, 15}. However, the time window for Task 46 ends at 3.84 hours after the start of the operations. The time window for Task 15 does not begin until 9.59 hours after the start. Therefore, it is infeasible for the tasks to be performed in the order {15, 46, 42}.

5.5.3 Scenario II: Closely Stationed Units

In the second scenario, Unit A and Unit B are located close to each other and the task locations are intermixed in the same area. The operations planning are done in the same way as in Scenario I. First, an operations plan is developed using decentralized planning. Then, an operations plan is developed using centralized operations planning.

Before the operations planning, the units generate the tasks. For this scenario, Unit A generated 25 of the tasks; Unit B generated the other 25. Each task can be performed at two locations each. The task locations are shown in Figure 5.13; the locations for a given task differ only in altitude. Unit A and Unit B each have three UAVs under their control for a total of six in the scenario. They each have a single UAV of Type 1 and two UAVs of Type 2 as described in Table 5.1.

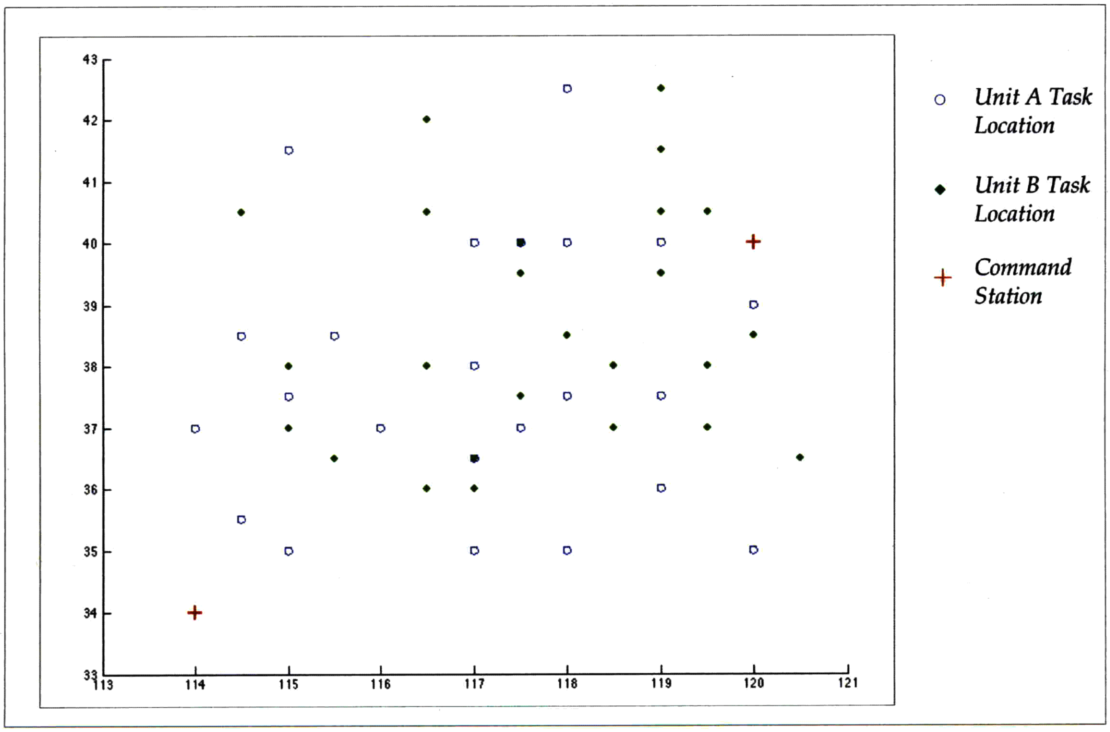


Figure 5.13 Scenario II Tasks

Both decentralized and centralized operations planning utilize COPA. For the decentralized planning, there are 36 composites generated for each unit, for a total of 72 composites. For centralized planning, there are 72 total composites generated.

5.5.3.1 Decentralized Operations Planning for Scenario II

In the decentralized operations planning for Scenario II, COPA was first used on to create an operations plan at the command center for Unit A. The tasks generated by Unit B were unknown to the algorithm during the planning. An operations plan was then created at the command center for Unit B.

The resulting operations plans are shown in Figure 5.14. The path plans are shown on separate graphs to make them easier to read; however, they are showing the same area. (The task locations are the same as in Figure 5.13.) As in Scenario I, the paths cross themselves due to the time window constraints. The operations plan for Unit A has a value of 905, while the operations plan for Unit B has a value of 907, for a total of 1812.

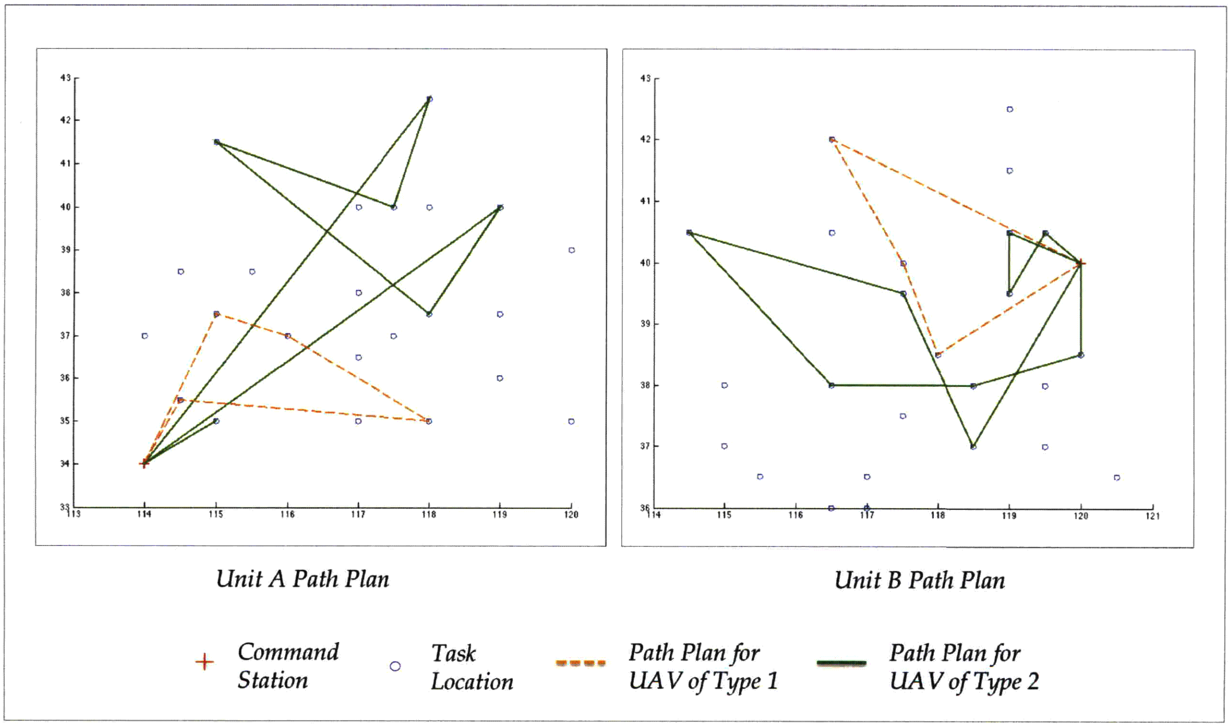


Figure 5.14 Decentralized Operations Plan for Scenario II

5.5.3.2 Centralized Operations Planning for Scenario II

For the centralized operations planning for Scenario II, the tasks from both units were input to the COPA software. A single run of the algorithm created an operations plan for the UAVs in both Unit A and Unit B. The results of the centralized operations planning are shown in Figure 5.15. The paths are shown on separate graphs to make them easier to read. The total value of the operations plan is 1884.

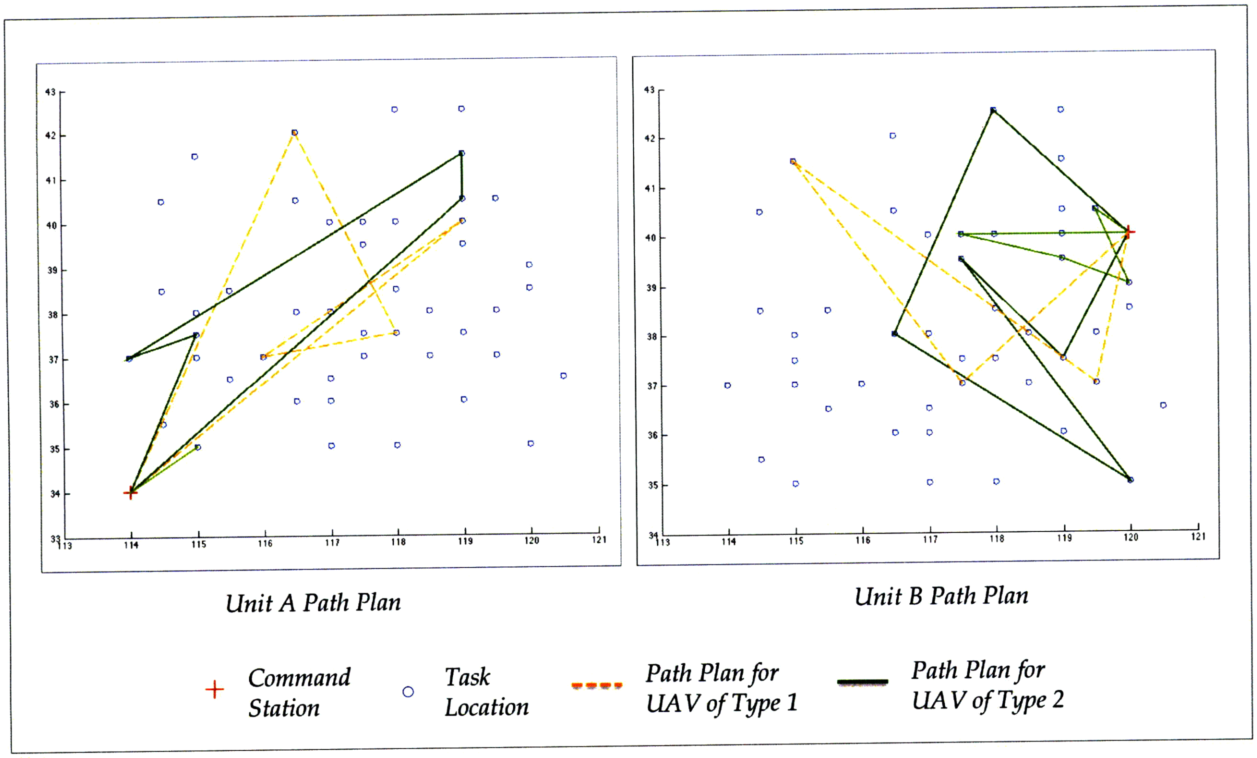


Figure 5.15 Centralized Operations Plan for Scenario II

5.5.3.3 Comparison between Decentralized and Centralized Planning for Scenario II

For Scenario II, the centralized operations plan is more valuable for the user. The value of the plan is 1884, while the total value of the decentralized operation plans is 1812, where the plan for Unit A's UAVs has a value of 905 and the plan for Unit B's UAVs has a value of 907.

In addition, more tasks were included in the centralized operations plan. In the decentralized operations plans, Unit A completed nine of its tasks and Unit B completed 12 of its tasks; therefore, a total of 21 tasks were included in the operations plan. In the centralized operations plan, 24 tasks were included, for a gain of 3 tasks.

The centralized planning resulted in a higher value of operations, because the UAVs are given the freedom to perform tasks for the other unit. In this scenario, UAVs from Unit A performed three of the tasks generated by Unit B. UAVs from Unit B performed seven of the tasks generated by Unit A.

For both centralized and decentralized operations planning, COPA developed valuable operations plans to the user. This illustrates that the algorithm can be used in either way to assist in planning for UAV operations.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 6

Conclusions and Future Work

The purpose of this chapter is to summarize the work presented in this thesis. The chapter first presents the contributions made through this thesis, including the methods and implementations developed to solve the UAV Planner Problem. The next section proposes possible modifications to the Composite Operations Planning Algorithm (COPA) and presents future research areas for the UAV Planner Problem. The end of the chapter summarizes the conclusions of this thesis.

6.1 Summary of Contributions

The purpose of this research is to develop an algorithm capable of solving the UAV Planner Problem that could be implemented for operational use. This thesis demonstrates the COPA's ability to solve the UAV Planner Problem. The contributions of this thesis are summarized in the following paragraphs.

- **Formulation and implementation of the Mixed-Integer Programming Model for the UAV Planner Problem.** Chapter 3 presents an MIP formulation that can provide an exact solution to the UAV Planner Problem. The MIP was implemented using OPL's ILOG Studio 5.5 and the CPLEX 11.0 solver.

- **Formulation and implementation of the Composite Operations Planning Algorithm (COPA).** COPA uses a metaheuristic in combination with a composite linear program to solve the UAV Planner Problem quickly. COPA is implemented in Java to create a software package that solves the UAV Planner Problem; this implementation is the COPA software. It is shown that the software can develop an operations plan for an instance of the UAV Planner Problem containing 15 heterogeneous UAVs and 500 tasks that can be performed at 10 locations each in less than 25 minutes.

- **Testing and comparison of MIP and COPA.** Chapter 5 presents the testing and comparison of the MIP and COPA, illustrating the strengths and weaknesses of each method. The MIP is able to provide an optimized operations plan for the UAV Planner Problem. However, instances with more than 15 tasks that could be performed at two locations each are intractable for the MIP. COPA provides operations plans that are valuable to the user, but might not find the most valuable solution. The runtimes for COPA are significantly shorter than the runtimes for the MIP implementation. COPA was able to provide operations plans for all instances of the UAV Planner Problem that were presented to the algorithm.

- **Operational scenarios to depict the use of COPA in operations planning.** Two scenarios are used to illustrate the use of COPA to fight wildfires. The scenario illustrates the use of the algorithm using both centralized and decentralized planning.

- **Recommendations for modifications to COPA**. The next section of this chapter proposes modifications to COPA that make it more likely to be able to execute operations plans developed by the algorithm.

6.2 Possible Modifications within Current Framework

While COPA incorporates the constraints of the UAV Planner Problem into the algorithm, there are additional operational constraints that could be incorporated. Adding these constraints to the algorithm would increase the capabilities of COPA in developing operations plans that are useful in an operational setting. Subsections 6.2.1 through 6.2.3 discuss possible modifications to COPA.

6.2.1 Incorporation of Weather Data

One of the concerns when planning UAV operations is the impact that weather will have on the observations. The presence of clouds, fog, or precipitation beneath the altitude from which an image is taken can make the image worthless to the user; the image will contain only the weather and not the desired location on the ground. Therefore, incorporating weather in to COPA can increase the affectivity of the operations plans created by the algorithm.

Although other weather data might be available, this section will focus on proposals to include the probability of cloud cover in the to the operations planning. The methods described for incorporating this data assume that the data is readily available to the algorithm.

One method to incorporate the probability of cloud cover in COPA is to use the data when deciding which composites to include in the operations plan. This can be done by altering the objective function of the FPCVF to balance the value of the composite with the probability of cloud cover. First, the probability that the tasks in the composite path will not be affected by cloud cover can be found by multiplying together the probabilities that each of the tasks included in the composite path will not be affected by cloud cover. Using $p_{i,k}$ to represent the probability of cloud cover for task i at location k and the set C to denote the set of task, location pairs in the composite path, the equation is:

$$\mathrm{P}(\textit{composite not affected by cloud cover}) = \omega_c = \prod_{(i,k)\in C}(1-p_{i,k}) \qquad (6.1)$$

The probability that cloud cover will not affect the composite is denoted by ω_c. Recall the notation in Section 4; γ_c is a binary decision variable representing the decision to include composite c in the operations plan and v_c denotes the value of composite c. The new objective function is:

$$\textit{Maximize} \qquad \sum_{c\in C}\omega_c v_c \gamma_c \qquad (6.2)$$

By multiplying the value of the composite by the probability that the composite will not be affected by weather, the composite value is scaled by this probability. Therefore, the composites with a high probability of cloud cover become less valuable. In this way, the objective function attempts to balance value and weather concerns.

A second method to include the probability of cloud cover in COPA would be to discount the value of tasks based on the probability that the task will not be affected by cloud cover before building the initial routes. For example, the value of a task could be replaced by the value of the task multiplied by one minus the probability of cloud cover, similar to the method above. In this method, the value of the tasks will be discounted throughout the algorithm.

There are other methods for incorporating more detailed weather information into COPA that would help create improved operations plans. Further research in this area would assist future UAV operations.

6.2.2 *Trajectory Planning Capabilities*

The current operations plans created by the COPA software do not include high fidelity trajectories between the tasks. Instead, the UAVs follow a straight-line trajectory between the task locations. Assuming the capability to create trajectories from other software, COPA can incorporate the trajectories into the operations planning. Because COPA relies on travel time to determine the operations plans, the travel time of an exact trajectory can replace the current travel times. The algorithm could then continue to build operations plans with the new matrix of travel times.

After the algorithm has produced an operations plan, the trajectories can be placed between the task locations to create exact trajectories for the UAVs.

6.2.3 *Entrance and Exit Locations for Tasks*

An *entrance location* is defined as the location where the UAV starts to perform a task. The *exit location* is the location where the UAV is at the conclusion of the task. Including entrance and exit locations for each task provides two additional capabilities.

First, it allows the operator to define partial paths to be included in the full path. For example, the operator could input that he or she would like tasks {1, 2, 3} to be performed in that order. In this case, tasks {1, 2, 3} would be treated as a single task with the entrance location at the location of Task 1 and the exit location at the location of Task 3. The observation length is the total time for the partial path.

Second, allowing the entrance and exit locations to differ would allow for the UAV to travel during the area tasks and exit at a separate location. This would improve the efficiency of the operations plans because the UAV could leave from the location in the area task that is closest to the next location in the path.

The entrance and exit locations for each task location could be incorporated into the Composite Generating step of COPA. When the paths are built, the algorithm uses the entrance location as the location that the UAV must reach to begin the task. The algorithm will use the exit location as the location where the UAV will travel from to the next path. It is assumed that the travel time between the entrance and exit locations are included in the observation time for the task.

6.3 Future Work

Future work will continue to increase the capability and effectiveness of operations planning for UAVs. In addition to the modifications proposed in the previous section, there are additional capabilities that would be beneficial to the operations planning process.

Two useful capabilities to consider are: (1) Synchronize UAVs to monitor an area or take an image of an area at the same time, and (2) Include heading information in the operations planning. Synchronization of UAVs at a particular task location can be beneficial to researchers and other users. The UAVs obtain aerial images of the same location at high and low resolutions and with different fields of view. Heading information is useful, because certain sun angles might cause images to become worthless to the user. By monitoring the heading and sun angle, we can increase the probability that a UAV collects high value data.

Additional future work could investigate alternative methods for solving the UAV Planner Problem. For example, the MIP formulation was solved using a branch-and-bound optimization method; however, another method might be able to solve the MIP in a shorter amount of time, such as approximate dynamic programming.

Alternatively, other heuristic methods could be used for the metaheuristic in COPA. Methods such as tabu search or simulated annealing could be used to build the full path composites needed for COPA. These methods might generate better composites for the algorithm, or the method might generate composites in a shorter runtime.

Another area of research includes applying the operations planning methods presented in this thesis to other applications. UAVs are currently being used for a wide variety of purposes: military, local law enforcement, and other surveillance operations. COPA could be used for the operations planning for the UAVs in these situations.

6.4 Conclusions

In summary, this work formulates the UAV Planner Problem for the purpose of creating operations plans for a set of UAVs to monitor Earth's phenomena. A mixed-integer program for the problem is formulated and implemented. In addition, an algorithm called COPA is presented for the operations planning for UAVs. COPA is implemented to create the COPA software. Through testing and analysis, it is shown that COPA can generate operations plans for instances of the UAV Planner Problem with up to 500 tasks that can be performed at 10 locations each.

We conclude that COPA is a viable algorithm for generating operations plans for UAVs. The algorithm provides operations plans that adhere to the constraints of the UAV Planner Problem. It allows for the input of a human operator in creating the plans. In addition, the algorithm provides a construct can be modified to include new capabilities, and will hopefully assist in future research on UAV operations planning.

APPENDIX A

GLOSSARY OF ACRONYMS

3PAA	Three Phase Approximate Algorithm
COPA	Composite Operations Planning Algorithm
EPOS	Earth Phenomena Observation System
ESTO	Earth Science Technology Office
FPCVBF	Full Path Composite Variable Binary Formulation
FPCVF	Full Path Composite Variable Formulation
LP	Linear Program
NASA	National Aeronautics and Space Administration
SBS	Space Based Sensor
UAV	Unmanned Aerial Vehicle
USFS	United States Forest Service
USV	Unmanned Surface Vehicle
USVOPP	USV Observation-Planning Problem

Computer Software Acronyms

CPLEX	Optimization software produced by ILOG
ILOG	Company owned by IBM which creates optimization software
JDK	Java Development Kit
OPL	Application created by OPL that uses CPLEX optimization software

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

COPA AND MIP COMPARISON TABLE

Size of Problem (Tasks, Locations)	Alg. Solution (24 Comps)	Alg. Runtime (s)	Alg. Solution (48 Comps)	Alg. Runtime (s)	Alg. Solution (72 Comps)	Alg. Runtime (s)	MIP Solution	MIP Runtime (s)	Lowest Optimality Gap	LP Relaxation Opt Value	LP Relaxation Run Time
(5,2)	245	0 81	255	1.17	255	1.376	295	2.93	13.56%	295	0.26
(5,2)	161	0 734	161	0.817	161	0.938	161	20 53	0%	201	0.25
(5,2)	274	0 613	275	0.526	275	0.848	281	7.42	2 14%	281	0 25
(5,2)	279	0.617	279	0.679	279	0.99	279	16.28	0%	344	0.25
(10,2)	543	0 357	543	0 98	543	0.913	555	3625	1.30%	670.25	1 5
(10,2)	660	0.348	660	0 832	664	0.939	739	2640.53	10 15%	756	1.5
(10,2)	634	0.398	696	0.456	696	0 49	696	6081	0%	764	1.32
(10,2)	432	0.323	434	0 563	434	0 866	466	2732.16	6 87%	558	1.81
(15,2)	851	0 82	888	1 099	888	1 163	959	7881 61**	2.10%	1007	28.53
(15,2)	884	0.825	884	0 941	884	1 088	884	1178.4**	0%	960	15:54.8
(15,2)	920	0.414	960	0.626	960	0.764	992	19866.41**	3.23%	1054	26.40 0
(15,2)	959	0 439	959	0.772	959	1.23	974	220.65	1.54%	1031	21:32.8
(20,2)	1207	0.764	1247	0 748	1265	0 989	703	4084.25**	***	1382	3.59.37.29
(20,2)	1102	0.457	1102	0.736	1102	1.537	784	2912.10**	***	1280	53.28
(20,2)	1138	0 543	1171	1.095	1190	1.211	1127	40092.43**	***	1330	11.13.04.44
(20,2)	1122	0 48	1125	0.757	1139	1.087	766	21816.79**	***	1240	35.53

* = approximate; computer ran out of memory in finding solution

** = time until solution was found--not able to prove, ran for atleast one more hour before terminating

*** = Algorithm solution greater than MIP solution

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

SCENARIO I: DECENTRALIZED OPERATIONS PLAN FOR UNIT A

*** COMPOSITE NO. 19
UAV Number: 1

Location	Start Time	End Time
Start Location	0	0
(14,2)	2.1363	3.7663
(20,1)	5.41	5.84
(3,1)	7.0331	8.0031
(9,1)	10.43	10.84

*** COMPOSITE NO. 21
UAV Number: 3

Location	Start Time	End Time
Start Location	0	1.8735
(15,2)	2.78	4.48
(24,2)	5.5574	6.9974
(1,2)	7.6573	7.8573
(27,2)	9.23	9.35
(2,1)	11.51	11.83

*** COMPOSITE NO. 35
UAV Number: 2

Location	Start Time	End Time
Start Location	0	4.3404
(7,1)	5.63	5.69
(12,1)	6.2888	9.1188
(26,2)	10.81	10.92

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D

SCENARIO I: DECENTRALIZED OPERATIONS PLAN FOR UNIT B

*** COMPOSITE NO. 1
UAV Number: 1

Location	Start Time	End Time
Start Location	0	0.1644
(16,1)	1.43	4.96
(19,2)	6.24	7.95
(20,1)	9.136	10.636

*** COMPOSITE NO. 18
UAV Number: 3

Location	Start Time	End Time
Start Location	0	0
(5,1)	1.0667	1.6467
(1,2)	3.32	3.69
(17,2)	6.99	7.34
(18,1)	8.65	8.66
(2,2)	9.3	9.59

*** COMPOSITE NO. 23
UAV Number: 2

Location	Start Time	End Time
Start Location	0	1.2562
(7,2)	2.37	2.45
(13,2)	5.1	6.92
(8,1)	10.64	10.67

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E

SCENARIO I: CENTRALIZED OPERATIONS PLAN

Unit A

*** COMPOSITE NO. 67
UAV Number: 1

Location	Start Time	End Time
Start Location	0	2.7686
(44,2)	4.76	6.2
(5,1)	7.02	7.3
(35,2)	8.7042	8.8642
(2,2)	11.51	11.83

*** COMPOSITE NO. 56
UAV Number: 2

Location	Start Time	End Time
Start Location	0	1.8735
(20,2)	2.78	4.48
(16,1)	4.7533	7.5833
(49,2)	9.23	9.35
(48,2)	10.81	10.92

*** COMPOSITE NO. 9
UAV Number: 3

Location	Start Time	End Time
Start Location	0	0
(18,2)	2.1363	3.7663
(1,2)	4.1683	4.3683
(9,2)	5.41	5.84
(4,1)	7.0331	8.0031
(11,1)	10.43	10.84

Unit B

*** COMPOSITE NO. 52
UAV Number: 4

Location	Start Time	End Time
Start Location	0	0
(19,1)	2.3198	2.8998
(32,2)	4.4784	5.9784
(39,2)	7.2296	7.5796
(23,1)	10.64	10.67

*** COMPOSITE NO. 47
UAV Number: 5

Location	Start Time	End Time
Start Location	0	1.2562
(22,2)	2.37	2.45
(13,1)	2.9	6.23
(40,1)	8.65	8.66
(10,2)	9.3	9.59

*** COMPOSITE NO. 60
UAV Number: 6

Location	Start Time	End Time
Start Location	0	2.3844
(46,2)	3.84	5.34
(42,2)	6.2625	7.9725
(15,1)	9.36	9.38

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F

SCENARIO II: DECENTRALIZED OPERATIONS PLAN FOR UNIT A

*** COMPOSITE NO. 2
UAV Type: 2

Location	Start Time	EndTime
Start Location	0	8.8842
(5,1)	10.78	10.79

**** COMPOSITE NO. 7
UAV Type: 1

Location	Start Time	EndTime
Start Location	0	3.2429
(15,2)	4.86	5.53
(19,1)	6.67	7.45
(1,2)	8.4432	9.2932
(12,2)	11.32	11.73

*** COMPOSITE NO. 12
UAV Type: 3

Location	Start Time	EndTime
Start Location	0	1.4264
(6,2)	4.32	4.53
(20,1)	5.0346	5.2046
(7,2)	7.1381	7.5281
(13,2)	8.82	9.23
(3,1)	10.33	10.35

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G

SCENARIO II: DECENTRALIZED OPERATIONS PLAN FOR UNIT B

*** COMPOSITE NO. 24
UAV Type: 3

Location	Start Time	EndTime
Start Location	0	0.2029
(3,2)	1.71	1.87
(1,1)	2.7887	2.9287
(17,2)	4.0289	4.9689
(10,2)	6.0956	7.4056
(5,1)	9.61	9.95
(8,2)	11.86	11.94

*** COMPOSITE NO. 31
UAV Type: 1

Location	Start Time	End Time
Start Location	0	0
(21,1)	1.213	1.843
(9,1)	7	9.36
(15,2)	11.75	11.85

*** COMPOSITE NO. 32
UAV Type: 2

Location	Start Time	End Time
Start Location	0	2.0098
(11,2)	3.23	6.32
(7,2)	7.51	8
(16,2)	11.45	11.47

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H

SCENARIO II: CENTRALIZED OPERATIONS PLAN

Unit A

*** COMPOSITE NO. 13
UAV Type: 1

Location	Start Time	End Time
Start Location	0	1.4264
(6,2)	4.32	4.53
(19,1)	6.67	7.45
(10,1)	8.2036	9.3436
(41,2)	11.45	11.47

*** COMPOSITE NO. 8
UAV Type: 2

Location	Start Time	End Time
Start Location	0	1.3224
(15,2)	4.86	5.53
(9,1)	7.57	7.94
(49,2)	13.0311	13.1311
(40,2)	13.8981	13.9981

*** COMPOSITE NO. 3
UAV Type: 3

Location	Start Time	EndTime
Start Location	0	8.8842
(5,1)	10.78	10.78

Unit B

*** COMPOSITE NO. 58
UAV Type: 4

Location	Start Time	End Time
Start Location	0	0
(46,1)	1.213	1.843
(20,1)	4.76	4.93
(34,1)	7	9.36
(13,2)	10.5758	10.9858

*** COMPOSITE NO. 53
UAV Type: 5

Location	Start Time	End Time
Start Location	0	0.2029
(28,2)	1.71	1.87
(26,1)	2.7887	2.9287
(4,2)	4.6504	5.9704
(35,2)	7.526	8.836
(3,1)	10.5019	10.5219

*** COMPOSITE NO. 54
UAV Type: 6

Location	Start Time	End Time
Start Location	0	0.4491
(25,1)	2.17	4.64
(7,2)	6.74	7.13
(23,2)	9.0264	9.1364
(27,2)	9.4643	9.6543
(30,1)	10.3031	10.6431
(33,2)	11.86	11.94

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] Applegate, D., Bixby, R. E., Chvatal, V., and Cook, W. J., *The Traveling Salesman Problem: A Computational Study,* Princeton University Press, Princeton, NJ, 2006.

[2] Armacost, A., "Composite Variables Formulations for Express Shipment Service Network Design," PhD Thesis in Operations Research, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[3] Armacost, A., Barnhart, C., and Ware, K., "Composite Variable Formulations for Express Shipment Service Network Design," *Transportation Science* 26, 2002.

[4] Baker, E. "An Exact Algorithm for the Time Constrained Traveling Salesman Problem," *Operations Research* 31(5), 938-945, 1983.

[5] Balas, E., "The Prize Collecting Traveling Salesman Problem," *Networks* 19, 621-636, 1989.

[6] Barth, C., "Composite Variable Formulation for Real-Time Mission Planning," Masters Thesis in Operations Research, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[7] Bellman, R., "Dynamic Programming Treatment of the Traveling Salesman Problem," *Journal of the Association for Computer Machinery* 9(1), 61-63, 1962.

[8] Bellmore, M., and Hong, S., "Transformation of Multisalesmen Problem to the Standard Traveling Salesman Problem," *Journal of the Association for Computer Machinery* 21, 500-504, 1974

[9] Bertsimas, D., and Tsitsiklis, J. N., *Introduction to Linear Optimization,* Athena Scientific, Belmont, MA, 1997.

[10] Blum, A, et al., "Approximation Algorithms for Orienteering and Discounted Reward TSP," *Society for Industrial and Applied Mathematics Journal of Computing* 37, 653-670, 2007.

[11] Boussier, S., Feillet, D., and Gendreau, M., "An Exact Algorithm for Team Orienteering Problems," *A Quarterly Journal of Operations Research* 5(3), 211-230, 2006.

[12] Chao, I., Golden, B, and Wasil, E, "A Fast and Effective Heuristic for the Orienteering Problem," *European Journal of Operational Research* 88, 475-489, 1996.

[13] Cohn, A., and Barnhart, C., "Improving Scheduling by Incorporating Key Maintenance Routing Decisions," *Operations Research* 51(3), 387-396, 2003.

[14] Cortez, P. and Morais, A., "A Data Mining Approach to Predict Forest Fire using Meteorological Data," *Portuguese Association for Artificial Intelligence,* 2007.

[15] Croes, G., "A Method for Solving Traveling-Salesman Problems," *Operations Research* 6(6), 791-812, 1958.

[16] Dantzig, G., Fulkerson, R., and Johnson, S., "Solution of a Large-Scale Traveling-Salesman Problem," *Operations Research* 2(4), 393-410, 1954.

[17] Delin, K. A., and Jackson, S. P., "A Sensor Web: A New Instrument Concept," *SPIE's Symposium on Integrated Optics*, 2001.

[18] Feillet, D., Dejax, P., and Gendreau, M., "Traveling Salesman Problem with Profits," *Transportation Science* 39(2), 188-205, 2005.

[19] Flood, M., "The Traveling Salesman Problem," *Operations Research* 4(1), 61-75, 1956.

[20] Gendreau, M., Hertz, A., Laporte, G., and Stan, M., "A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows," *Operations Research* 46(3), 330-335, 1998.

[21] Geomens, M. X., and Williamson, D. P., "A General Approximation Technique for Constrained Forest Problems," *Society for Industrial and Applied Mathematics Journal of Computing* 24(2), 296-317, 1992.

[22] Gillett, B. E., and Miller, L. R., "A Heuristic Algorithm for the Vehicle-Dispatching Problem," *Operations Research* 22(2), 340-349, 1974.

[23] Golden, B. L., Levy, L., and Vohra, R., "The Orienteering Problem," *Naval Research Logistics* 37, 307-318, 1987.

[24] Golden, B., Wang, Q., Liu, L., "A Multifaceted Heuristic for the Orienteering Problem," *Naval Research Logistics* 35 (3), 359-366, 1988.

[25] Hillier, F. S. and Lieberman, G. J., *Introduction to Operations Research,* McGraw-Hill Professional, 2004.

[26] Kantor, M.G. and Rosenwein, M. B., "The Orienteering Problem with Time Windows," *The Journal of the Operational Research Society* 43(6), 629-635, 1992.

[27] Laporte, G., "The Traveling Salesman Problem: Overview of Algorithms," *European Journal of Operational Research* 59(2), 231-247, 1992.

[28] Laporte, G. and Martello, S., "The Selective Traveling Salesman Problem," *Discrete Applied Mathematics* 26, 193-207, 1990.

[29] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A., and Shmoys, D. B., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* John Wiley and Sons, Inc, New York, 1985.

[30] Miller, J. V., "Large-Scale Dynamic Observation Planning for Unmanned Surface Vessels," Masters Thesis, Massachusetts Institute of Technology, 2007.

[31] "NASA and Forest Service Partner on California Wildfires," *USDA Forest Service News Release*, 17 August 2007, <www.fs.fed.us/r5/news/2007/nasa-fs.shtml>.

[32] "NASA Images of Wildfires," *NASA/U.S. Forest Service*, 5 November 2007, <http://www.nasa.gov/vision/earth/lookingatearth/socal_wildfires_oct07.html>.

[33] Nielsen, C., "Large-Scale Network Design using Composite Variables: An Application for Air Mobility Command's 30-day Channel Route Network," Masters Thesis in Operations Research, Massachusetts Institute of Technology, Cambridge, MA, 2002.

[34] Ramesh, R. and Brown, K. M., "An Efficient Four-Phase Heuristic for the Generalized Orienteering Problem," *Computers and Operations Research* 18, 151-165, 1991.

[35] Ramesh, R., Yoon, Y., and Karwan, M. H., "An Optimal Algorithm for the Orienteering Tour Problem," *ORSA Journal on Computing* 4, 1992.

[36] Righini, G. and Salani, M., "Decremental State Space Relaxation Strategies and Initialization Heuristics for Solving the Orienteering Problem with Time Windows with Dynamic Programming," *Computers and Operations Research* 36, 1191-1203, 2009.

[37] Ropke, S., Cordeau, J., and Laporte, G.. "Models and Branch-and-Cut Algorithms for Pickup and Delivery Problem with Time Windows," *Networks* 49(4), 258-272, 2007.

[38] Rosenkrantz, D., Stearns, R., and Lewis, P., "An Analysis of Several Heuristics for the Traveling Salesman Problem," *Society of Industrial and Applied Mathematics Journal of Computing* 6(3), 563-581, 1977.

[39] Russell, R. A., "M-Tour Traveling Salesman Problem," *Operations Research* 25, 517-524, 1977.

[40] "Sensor Web Dynamic Replanning," *NASA ESTO AIST-05 Year 1 Annual Review*, September 5, 2007.

[41] Talabec, S. J., "Sensor Webs: An Emerging Concept for Future NASA Systems," *An Information Systems Center Technology and Assessment Seminar*, 16 May 2002, Powerpoint Slideshow, 16.

[42] Tsiligrides, T. "Heuristic Methods Applied to Orienteering." *Journal of the Operational Research Society* 35, 797-809, 1984.

[43] Vansttenwegen, P., Souffriau, W., Vanden Berghe, G., and van Oudheusen, D., "A Guided Local Search Heuristic for the Team Orienteering Problem," *European Journal of Operational Research* 196, 118-127, 2009.